PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-040090
(43)Date of publication of application : 13.02.1998

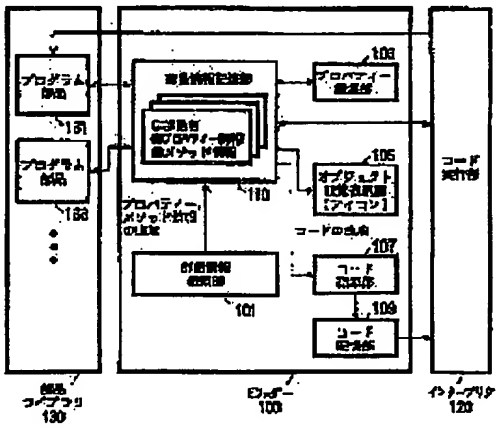
(51)Int.Cl. G06F 9/06

(21)Application number : 08-171998
(22)Date of filing : 02.07.1996

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : HAMADA SEIJI
NAKAMURA YUICHI
TAGO KAZUYA

(54) PROGRAM DEVELOPMENT SUPPORTING SYSTEM, ITS METHOD AND STORAGE MEDIUM FOR STORING PROGRAM COMPONENT TO BE USED FOR PROGRAM DEVELOPMENT SUPPORT

(57)Abstract:
PROBLEM TO BE SOLVED: To obtain various programming desired for a programmer by using program components.
SOLUTION: The apparent function of a program component 131 is extended by a component information editing part 101 for updating information stored in a component information storing part 110 for storing information related to the property and method of the program component 131. Component information is used for both of the development and execution of an application program. When the information is used for a property edition part 103 and a code edition part 107 in developing the program, a user can utilize the function of the component for the generation of a programming code as if the function is extended and it is unnecessary to be conscious of a difference between the function of the existing program component and the extended function. When executing the program, the information is used for accessing the component 131 from an interpreter 120 and the extended function is properly executed.



LEGAL STATUS

[Date of request for examination]	22.12.1999
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3427918
[Date of registration]	16.05.2003

[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平10－40090
(43)公開日　平成10年(1998)２月13日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/06	５３０		Ｇ０６Ｆ　9/06	５３０Ｗ ５３０Ｐ

審査請求　未請求　請求項の数12　ＯＬ　（全 28 頁）

(21)出願番号　特願平8－171998

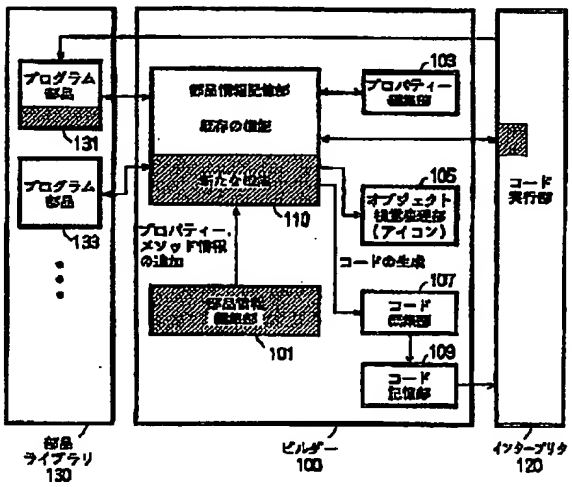
(22)出願日　平成８年(1996)７月２日

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）
(72)発明者　猪田　誠司
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社　大和事業所内
(74)代理人　弁理士　合田　潔　（外２名）

最終頁に続く

(54)【発明の名称】　プログラム開発支援システム及び支援方法、プログラム開発支援のために用いられるプログラム部品を格納する記憶媒体

(57)【要約】
【課題】　プログラマが欲する種々多様なプログラミングをプログラム部品を使用して可能にする。
【解決手段】　プログラム部品のプロパティとメソッドに関する情報を保持する部品情報記憶部110の情報を更新する部品情報編集部101により、プログラム部品131の見かけ上の機能を拡張する。部品情報はアプリケーション・プログラム開発時と実行時の両方に利用される。プログラム開発時には、プロパティ編集部103、コード編集部107において利用することにより、ユーザは部品の機能があたかも拡張されたような感覚でプログラミング・コードの作成に利用することができ、しかも既存のプログラム部品の機能と拡張された機能の違いを意識することはない。一方、実行時には、インタプリタ120からプログラム部品131を呼び出すために利用され、拡張された機能を適宜実行することができる。



1

【特許請求の範囲】

【請求項1】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するメソッドとプロパティの情報を提供するプログラム部品を含むデータ処理システムであって、

（a）ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューを管理する部品プロパティテーブルと、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることにより、ユーザ定義メソッドのコードをポイントするメソッドコールアダプタを含むプログラム部品と、

（b）プログラム部品名と、ユーザ定義プロパティ名及びジェネリック・プロパティ・アクセス関数を管理するプロパティテーブルと、ユーザ定義メソッド名及びジェネリック・メソッド関数を管理するメソッド・テーブルを備える部品情報記憶部と、

（c）前記プロパティ・テーブルのユーザ定義プロパティ名の登録及び前記メソッド・テーブルのユーザ定義メソッド名の登録を行うための部品編集部と

（d）前記プログラム部品のユーザ定義プロパティ・バリューを更新するためのプロパティ編集部と、

（e）メソッドの実行内容を特定するメソッドのコードを編集するため、および、部品名とプロパティ名とメソッド名の指定を含むメインロジックを編集するためのコード編集部と、

（f）前記コード編集部で編集されたメソッドのコードを部品名とメソッド名によって特定可能に格納し、前記メインロジックを格納するコード記憶部と、

（g）前記コード記憶部に格納されたメインロジックを受領し、前記メインロジックに含まれる部品名とプロパティ名から、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定し、前記プログラム部品に対し、ユーザ定義プロパティ名とジェネリック・プロパティ・アクセス関数を発行し、前記メインロジックに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定し、前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行するコード実行部と、

を含むシステム。

【請求項2】（a）ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューを管理する部品プロパティテーブルを含むプログラム部品と、

（b）前記プログラム部品のユーザ定義プロパティ・バリューにアクセスするためのユーザ定義プロパティ名と、ジェネリック・プロパティ・アクセス関数を管理するプロパティテーブルを備える部品情報記憶部と、

（c）前記プロパティ・テーブルのユーザ定義プロパティ名と前記部品プロパティ・テーブルのユーザ定義プロパティ名を登録するための部品編集部とを含むプログラム開発支援システム。

2

【請求項3】（a）ユーザ定義メソッド名とジェネリック・メソッド関数を受けることにより、ユーザ定義メソッドのコードをポイントするメソッドコールアダプタを含むプログラム部品と、

（b）プログラム部品名とユーザ定義メソッド名と、ジェネリック・メソッド関数を管理するメソッド・テーブルを備える部品情報記憶部と、

（c）前記メソッド・テーブルのユーザ定義メソッド名を登録するための部品編集部と

（d）メソッドの実行内容を特定するメソッドのコードを編集するため、および、部品名とメソッド名の指定を含むメインロジックを編集するためのコード編集部と、

（e）前記コード編集部で編集されたメソッドのコードを部品名とメソッド名によって特定可能に格納し、前記メインロジックを格納するコード記憶部と、

（f）前記コード記憶部に格納されたメインロジックを受領し、前記メインロジックに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定し、前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行するコード実行部と、

を含むシステム。

【請求項4】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するメソッドとプロパティを提供するプログラム部品と、前記プログラム部品のプロパティ名、プロパティ・アクセス関数を管理するプロパティ・テーブル及び、前記プログラム部品のメソッド名、実装関数アドレス情報を管理するメソッド・テーブルとを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法であって、

（a）ユーザによるプログラム部品の特定を検出する段階と、

（b）前記特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリとメソッド名を入力するためのメソッド情報エントリを表示する段階と、

（c）前記プロパティ情報エントリに入力されたユーザ定義プロパティ名を前記プロパティ・テーブルに登録する段階と、

（d）前記メソッド情報エントリに入力されたユーザ定義メソッド名を前記メソッド・テーブルに登録する段階と、

を含む方法。

【請求項5】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するプロパティ・バリューを提供するプログラム部品とを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法であって、

ッド・テーブルと、コード実行部と、プログラム部品名、プロパティ名、メソッド名の記述を含み、前記コード実行部で実行するためのプログラム・コードと前記プログラム部品のメソッドに対応付けられて格納されたユーザ定義メソッドのコードを記憶するコード記憶部と、前記コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法であって、

（a）プログラム部品名と前記プログラム部品のプロパティ名と前記プログラム部品のメソッド名を指定する記述を含むプログラムコードを受領する段階と、

（b）前記プログラムコードに含まれる部品名とプロパティ名から、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定する段階と、

（c）前記プログラム部品名と、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいて前記プログラム部品のプロパティ・バリューを特定する段階と、

（d）前記プログラムコードに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定する段階と、

（e）前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行する段階と、

（f）プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領する段階と、

（g）前記ジェネリック・メソッド・コード・アクセス情報を基づいて、前記コード記憶部に格納されたユーザ定義メソッドのコードにアクセスする段階と、

（h）前記ユーザ定義メソッドのコードを実行する段階と、

を含む方法。

【請求項8】プロパティ情報を管理するプログラム部品と、前記プログラム部品のプロパティ名、ジェネリック・プロパティ・アクセス関数を管理するプロパティ・テーブルと、コード実行部と、プログラム部品名、プロパティ名の記述を含み、前記コード実行部で実行するためのプログラム・コードを記憶するコード記憶部と、前記コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法であって、

（a）プログラム部品名と前記プログラム部品のプロパティ名を指定する記述を含むプログラムコードを受領する段階と、

（b）前記プログラムコードに含まれる部品名とプロパティ名から、前記部品情報記憶部のプロパティ・テーブ

（a）ユーザによる新規プログラム部品の登録のためのプログラム部品を象徴する新規プログラム部品のアイコンを前記表示装置に表示する段階と、

（b）ユーザによる新規プログラム部品のアイコンの特定を検出する段階と、

（c）前記特定された新規プログラム部品のプログラム部品名を入力するためのプログラム部品名入力エントリを表示する段階と、

（d）前記プログラム部品入力エントリに入力されたユーザ定義プログラム部品名に対応したプロパティ名及びプロパティ・バリューのエントリを有する部品プロパティ・テーブルを作成する段階と、

（e）前記特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリを表示する段階と、

（f）前記プロパティ情報エントリに入力されたユーザ定義プロパティ名を前記部品プロパティ・テーブルに登録する段階と、

（g）前記ユーザ定義プロパティ名に対応するプロパティ・バリューを入力するためのプロパティ・バリュー入力エントリを表示する段階と、

（h）前記プロパティ・バリュー入力エントリに入力された値を前記部品プロパティ・テーブルに登録する段階と、

を含む方法。

【請求項6】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するプロパティ・バリューを提供するプログラム部品とを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法であって、

（a）ユーザによるプログラム部品の特定を検出する段階と、

（b）前記特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリを表示する段階と、

（c）前記プロパティ情報エントリに入力されたユーザ定義プロパティ名を前記特定されたプログラム部品と関連付ける段階と、

（d）前記ユーザ定義プロパティ名に対応するプロパティ・バリューを入力するためのプロパティ・バリュー入力エントリを表示する段階と、

（e）前記プロパティ・バリュー入力エントリに入力された値を前記特定されたプログラム部品に関連付ける段階と、

を含む方法。

【請求項7】メソッド情報とプロパティ情報を管理するプログラム部品と、前記プログラム部品のプロパティ名、ジェネリック・プロパティ・アクセス関数を管理するプロパティ・テーブル及び、前記プログラム部品のメソッド名、ジェネリック・メソッド関数を管理するメソ

5

ルで管理されているジェネリック・プロパティ・アクセス関数を特定する段階と、
（ｃ）前記プログラム部品名と、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいて前記プログラム部品のプロパティ・バリューを特定する段階と、
を含む方法。
【請求項９】メソッド情報を管理するプログラム部品と、前記プログラム部品のメソッド名、ジェネリック・メソッド関数を管理するメソッド・テーブルと、コード実行部と、プログラム部品名、メソッド名の記述を含み、前記コード実行部で実行するためのプログラム・コードと前記プログラム部品のメソッドに対応付けられて格納されたユーザ定義メソッドのコードを記憶するコード記憶部と、前記コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法であって、
（ａ）プログラム部品名と前記プログラム部品のメソッド名を指定する記述を含むプログラムコードを受領する段階と、
（ｂ）前記プログラムコードに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定する段階と、
（ｃ）前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行する段階と、
（ｄ）プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領する段階と、
（ｅ）前記ジェネリック・メソッド・コード・アクセス情報を基づいて、前記コード記憶部に格納されたユーザ定義メソッドのコードにアクセスする段階と、
（ｆ）前記ユーザ定義メソッドのコードを実行する段階と、
を含む方法。
【請求項１０】実行時において、自ら管理するメソッドとプロパティを提供し、プログラム開発支援のためにプログラム開発支援システムにロードされるプログラム部品を格納する記憶媒体であって、
該プログラム部品は、
（ａ）ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューのエントリを有する部品プロパティテーブルと、
（ｂ）ユーザ定義メソッド名とジェネリック・メソッド関数を前記プログラム開発支援システムから受けることにより、メソッドのコードをポイントするメソッドコールアダプタと、
を含むことを特徴とする記憶媒体。

6

【請求項１１】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するメソッドとプロパティを提供するプログラム部品と、前記プログラム部品のプロパティ名、プロパティ・アクセス関数を管理するプロパティ・テーブル及び、前記プログラム部品のメソッド名、実装関数アドレス情報を管理するメソッド・テーブルとを含むデータ処理システムにおいて、プログラム部品の機能を拡張するためのプログラムを格納する記憶媒体であって、
該プログラムは、
（ａ）ユーザによるプログラム部品の特定を検出することを前記データ処理システムに指示するプログラムコード手段と、
（ｂ）前記特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリとメソッド名を入力するためのメソッド情報エントリを表示することを前記データ処理システムに指示するプログラムコード手段と、
（ｃ）前記プロパティ情報エントリに入力されたユーザ定義プロパティ名を前記プロパティ・テーブルに登録することを前記データ処理システムに指示するプログラムコード手段と、
（ｄ）前記メソッド情報エントリに入力されたユーザ定義メソッド名を前記メソッド・テーブルに登録することを前記データ処理システムに指示するプログラムコード手段と、
を含む記憶媒体。
【請求項１２】メソッド情報とプロパティ情報を管理するプログラム部品と、前記プログラム部品のプロパティ名、ジェネリック・プロパティ・アクセス関数を管理するプロパティ・テーブル及び、前記プログラム部品のメソッド名、ジェネリック・メソッド関数を管理するメソッド・テーブルと、コード実行部と、プログラム部品名、プロパティ名、メソッド名の記述を含み、前記コード実行部で実行するためのプログラム・コードと前記プログラム部品のメソッドに対応付けられて格納されたユーザ定義メソッドのコードを記憶するコード記憶部と、前記コードを編集するためのコード編集部とを含むデータ処理システムにおいて、前記プログラム・コードを実行するためのプログラムを格納する記憶媒体であって、
該プログラムは、
（ａ）プログラム部品名と前記プログラム部品のプロパティ名と前記プログラム部品のメソッド名を指定する記述を含むプログラムコードを受領することを前記データ処理システムに指示するプログラムコード手段と、
（ｂ）前記プログラムコードに含まれる部品名とプロパティ名から、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定することを前記データ処理システムに指示するプログラムコード手段と、

（ｃ）前記プログラム部品名と、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいて前記プログラム部品のプロパティ・バリューを特定することを前記データ処理システムに指示するプログラムコード手段と、
（ｄ）前記プログラムコードに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定することを前記データ処理システムに指示するプログラムコード手段と、
（ｅ）前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行することを前記データ処理システムに指示するプログラムコード手段と、
（ｆ）プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領することを前記データ処理システムに指示するプログラムコード手段と、
（ｇ）前記ジェネリック・メソッド・コード・アクセス情報を基づいて、前記コード記憶部に格納されたユーザ定義メソッドのコードにアクセスすることを前記データ処理システムに指示するプログラムコード手段と、
（ｈ）前記ユーザ定義メソッドのコードを実行することを前記データ処理システムに指示するプログラムコード手段と、
を含む記憶媒体。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】この発明は、プログラムの作成を支援する方法及びシステムに関し、詳しくは、プログラミング構築(開発)支援装置にプログラム部品を統合する方法に関し、さらに詳しくは、プログラム部品が本来持っている機能に加え、新たな機能を追加しながら統合化する方法に関する。

【０００２】

【従来の技術】従来より、Visual Basic（"Visual Basic "は、マイクロソフト社の商標)等の対話型プログラム作成支援システムにおいて、プログラマがアプリケーションプログラムを対話的に作成する際、その作成を支援するため、プログラム部品を提供している。図１５は、従来のプログラム部品の利用形態を示す図である。

【０００３】この例では、ボタン・アイコン２０３のプログラム部品２３０が提供されており、プロパティ２３１として、ボタン・アイコンを表示する位置を定義するための変数情報（ｘ，ｙ）と、ボタン・アイコンの幅と高さ定義するための変数情報（ｗ．ｈ）と、メッソド２３３として、指定されたｘ，ｙの位置にボタン・アイコンを移動することを指示する「ｍｏｖｅ（ｘ，ｙ）」と、指定されたｗ，ｈの大きさにボタン・アイコンを変更する「ｒｅｓｉｚｅ（ｗ，ｈ）」等の情報を含んでいる。

【０００４】このプログラム部品がアプリケーション・ウインドウに位置づけられ、後述するプロパティ編集部でプロパティの値が設定されると、インタプリタはこれらの情報を解釈し、アプリケーション・ウインドウ上で、設定されたプロパティ・バリューに従ったボタン・アイコンが表示されることとなる。

【０００５】しかし、このような従来の技術におけるプログラム部品においては、プロパティやメソッドは予め与えられており、利用できるプロパティ、メソッドは限定されたものであった。従って、ボタンの部品に、「ラベルを複数付けたい（従来のプログラミング支援システムにおいて、プログラム部品には、通常、ボタンに１つのラベルしか付すことができない）」、「そのボタンに付されるラベルの文字数に対応させてボタンの大きさを変更したい（従来のプログラミング支援システムにおいて、プログラム部品には、通常、定数値によってサイズを定義し、関数値によってはサイズを定義できない）」といった多種多様なプログラマの要求を満足させることはできず、プログラム部品を利用したプログラミングには多くの制約が存在していた。

【０００６】また、従来のプログラム部品は、ボタン・アイコンや、ルーラ・バー等の視覚的に図形として表示されるグラフィカル・オブジェクトに対してのみ提供されており、後述するコード編集部で文字や記号を用いて記述される非グラフィカル・オブジェクトに対して提供されていなかった。このため、プログラマは、コーディングを行うプログラミング言語の命令を複数つなぎ合わせ、記述することによりプログラムを作成する必要があった。また、他のプログラムの作成において、かかる記述を再利用することはできず、再度ほぼ同様のコーディングを行う必要があった。サブルーチンを用いたコーディングにおいても、各サブルーチンは、プログラム部品のようなステートを保有できないため、オブジェクト固有の変数の管理が複雑になり、また、その再利用は容易ではない。

【０００７】かかる従来の問題に関連して、特開平７－２８６３６号公報は、入力操作と視覚オブジェクトを対応付けたテーブルを用いてグラフィカル・ユーザ・インターフェース構築支援装置を提案している。しかし、ここで提案されているグラフィカル・ユーザ・インターフェース構築支援装置は、アプリケーション・プログラム内の利用者による各入力操作に対応する視覚オブジェクトが、その入力操作にどの程度適合しているかを示すため、視覚オブジェクトの候補をその入力操作への適合度の高い順に呈示するものであり、プログラム部品の選択を迅速に行うことはできるが、プログラム部品の機能自体を拡張することはできない。

【０００８】また、特開平７－２１９７５３号公報は、ボタン・アイコンがポインティングデバイスのポインタで押された時等に発生するイベントに対し実行する動作を指示するといったコールバック関数の一部をリソース化することによって、開発効率の向上を行う会話型プログラム開発支援システムを提案している。この方式は、メソッドによって実行される処理内容を変更することができるが、新たなメソッドを追加することはできないため、「そのボタンに付されるラベルの文字数に対応させてボタンの大きさを変更したい」といったプログラマの多種多様な要求を満足させることができない。

【０００９】さらに、特開平４－３５３９２６号公報は、会話部品定義体入力操作と視覚オブジェクトをテーブルを用いて対応付ける方式を提案している。しかし、この方式は、会話部品に予めメソッドが対応付けられており、そのメソッドをカスタマイズするものであり、会話部品にユーザの欲するメソッドを追加することはできない。このため、「ラベルを複数付けたい」、「そのボタンに付されるラベルの文字数に対応させてボタンの大きさを変更したい」といった多種多様なプログラマの要求を満足させることはできず、また、非グラフィカル・オブジェクトをプログラム部品として定義することができなかった。

【００１０】

【発明が解決しようとする課題】この発明の目的は、部品情報記憶部において、プログラム部品が本来持っている機能に関する情報を蓄積するだけでなく、実際の部品にはない新たな機能も蓄積し、プログラムビルダやインタプリタに利用可能にすることにより、従来のプログラム部品では実現できなかったプログラマが欲する種々多様なプログラミングをプログラム部品を利用しつつ実現することである。

【００１１】本発明の他の目的は、プログラム部品が元々持っていたプロパティと、新たに追加されたプロパティをプロパティ編集部で区別なく変更可能にすることである。

【００１２】本発明の他の目的は、新たに追加されたメソッドやプロパティの利用に際して、プログラム部品が元々持っていたものと区別なくプログラムビルダやインタプリタで利用可能にすることである。

【００１３】

【課題を解決するための手段】本発明のアプリケーションプログラム支援システムは、プログラム部品に関する情報を保持する部品情報記憶部を有しており、この情報を更新することにより、プログラム部品の実際の機能を拡張することなく、見かけ上の機能を拡張する。部品情報の更新は部品に新たなプロパティならびにメソッドを追加することに対応しており、このような編集作業は部品情報編集部により行われる。部品情報はアプリケーション・プログラム開発時と実行時の両方に利用される。

すなわち、プログラム開発時には、プロパティ編集部、コード編集部において利用することにより、ユーザは部品の機能があたかも拡張されたような感覚でプログラミング・コードの作成に利用することができ、しかも通常のプログラム部品（ユーザが定義したプログラム部品ではなく、固定的な機能を有するプログラム部品）本来の機能と拡張された機能の違いを意識することはない。一方、実行時には、インタプリターからプログラム部品を呼び出すために利用され、拡張された機能を適宜実行することができる。

【００１４】本発明の１の態様においては、ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューを管理する部品プロパティテーブルと、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることにより、ユーザ定義メソッドのコードをポイントするメソッドコールアダプタを含むプログラム部品と、プログラム部品名と、ユーザ定義プロパティ名及びジェネリック・プロパティ・アクセス関数を管理するプロパティテーブルと、ユーザ定義メソッド名及びジェネリック・メソッド関数を管理するメソッド・テーブルを備える部品情報記憶部と、プロパティ・テーブルのユーザ定義プロパティ名の登録及びメソッド・テーブルのユーザ定義メソッド名の登録を行うための部品編集部とプログラム部品のユーザ定義プロパティ・バリューを更新するためのプロパティ編集部と、メソッドの実行内容を特定するメソッドのコードを編集するため、および、部品名とプロパティ名とメソッド名の指定を含むメインロジックを編集するためのコード編集部と、コード編集部で編集されたメソッドのコードを部品名とメソッド名によって特定可能に格納し、メインロジックを格納するコード記憶部と、コード記憶部に格納されたメインロジックを受領し、メインロジックに含まれる部品名とプロパティ名から、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定し、プログラム部品に対し、ユーザ定義プロパティ名とジェネリック・プロパティ・アクセス関数を発行し、メインロジックに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定し、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行するコード実行部とを含むシステムが提供される。

【００１５】本発明の他の１態様においては、ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューを管理する部品プロパティテーブルを含むプログラム部品と、プログラム部品のユーザ定義プロパティ・バリューにアクセスするためのユーザ定義プロパティ名と、ジェネリック・プロパティ・アクセス関数を管理するプロパティテーブルを備える部品情報記憶部と、プロパティ・テーブルのユーザ定義プロパティ

名と部品プロパティ・テーブルのユーザ定義プロパティ名を登録するための部品編集部とを含むプログラム開発支援システムが提供される。

【００１６】本発明の他の１態様においては、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることにより、ユーザ定義メソッドのコードをポイントするメソッドコールアダプタを含むプログラム部品と、プログラム部品名とユーザ定義メソッド名と、ジェネリック・メソッド関数を管理するメソッド・テーブルを備える部品情報記憶部と、メソッド・テーブルのユーザ定義メソッド名を登録するための部品編集部と、メソッドの実行内容を特定するメソッドのコードを編集するため、および、部品名とメソッド名の指定を含むメインロジックを編集するためのコード編集部と、コード編集部で編集されたメソッドのコードを部品名とメソッド名によって特定可能に格納し、メインロジックを格納するコード記憶部と、コード記憶部に格納されたメインロジックを受領し、メインロジックに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定し、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行するコード実行部とを含むシステムが提供される。

【００１７】本発明の他の１態様においては、ユーザによるプログラム部品の特定を検出する段階と、特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリとメソッド名を入力するためのメソッド情報エントリを表示する段階と、プロパティ情報エントリに入力されたユーザ定義プロパティ名をプロパティ・テーブルに登録する段階と、メソッド情報エントリに入力されたユーザ定義メソッド名をメソッド・テーブルに登録する段階と、を含むプログラム部品の機能を拡張する方法が提供される。

【００１８】本発明の他の１態様においては、ユーザによる新規プログラム部品の登録のためのプログラム部品を象徴する新規プログラム部品のアイコンを表示装置に表示する段階と、ユーザによる新規プログラム部品のアイコンの特定を検出する段階と、特定された新規プログラム部品のプログラム部品名を入力するためのプログラム部品名入力エントリを表示する段階と、プログラム部品入力エントリに入力されたユーザ定義プログラム部品名に対応したプロパティ名及びプロパティ・バリューのエントリを有する部品プロパティ・テーブルを作成する段階と、特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリを表示する段階と、プロパティ情報エントリに入力された情報を部品プロパティ・テーブルに登録する段階と、ユーザ定義プロパティ名に対応するプロパティ・バリューを入力するためのプロパティ・バリュー入力エントリを表示する段階と、プロパティ・バリュー入力エントリに入力された値を部品プロパティ・テーブルに登録する段階とを含む入力装置と、表示装置と、コード実行部と、コード実行部に自ら管理するプロパティ・バリューを提供するプログラム部品とを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法が提供される。

【００１９】本発明の他の１態様においては、ユーザによるプログラム部品の特定を検出する段階と、特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリを表示する段階と、プロパティ情報エントリに入力されたユーザ定義プロパティ名を特定されたプログラム部品と関連付ける段階と、ユーザ定義プロパティ名に対応するプロパティ・バリューを入力するためのプロパティ・バリュー入力エントリを表示する段階と、プロパティ・バリュー入力エントリに入力された値を特定されたプログラム部品に関連付ける段階とを含む入力装置と、表示装置と、コード実行部と、コード実行部に自ら管理するプロパティ・バリューを提供するプログラム部品とを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法が提供される。

【００２０】本発明の他の１態様においては、プログラム部品名とプログラム部品のプロパティ名とプログラム部品のメソッド名を指定する記述を含むプログラムコードを受領する段階と、プログラムコードに含まれる部品名とプロパティ名から、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定する段階と、プログラム部品名と、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいてプログラム部品のプロパティ・バリューを特定する段階と、プログラムコードに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定する段階と、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行する段階と、プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領する段階と、ジェネリック・メソッド・コード・アクセス情報を基づいて、コード記憶部に格納されたユーザ定義メソッドのコードにアクセスする段階と、ユーザ定義メソッドのコードを実行する段階とを含むプログラム・コードの実行方法が提供される。

【００２１】本発明の他の１態様においては、プログラム部品名とプログラム部品のプロパティ名を指定する記述を含むプログラムコードを受領する段階と、プログラムコードに含まれる部品名とプロパティ名から、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定する段階と、プログラム部品名と、部品情報記憶部のプロパティ

・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいてプログラム部品のプロパティ・バリューを特定する段階とを含むプロパティ情報を管理するプログラム部品と、プログラム部品のプロパティ名、ジェネリック・プロパティ・アクセス関数を管理するプロパティ・テーブルと、コード実行部と、プログラム部品名、プロパティ名の記述を含み、コード実行部で実行するためのプログラム・コードを記憶するコード記憶部と、コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法が提供される。

【００２２】本発明の他の１態様においては、プログラム部品名とプログラム部品のメソッド名を指定する記述を含むプログラムコードを受領する段階と、プログラムコードに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定する段階と、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行する段階と、プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領する段階と、ジェネリック・メソッド・コード・アクセス情報を基づいて、コード記憶部に格納されたユーザ定義メソッドのコードにアクセスする段階と、ユーザ定義メソッドのコードを実行する段階とを含むメソッド情報を管理するプログラム部品と、プログラム部品のメソッド名、ジェネリック・メソッド関数を管理するメソッド・テーブルと、コード実行部と、プログラム部品名、メソッド名の記述を含み、コード実行部で実行するためのプログラム・コードとプログラム部品のメソッドに対応付けられて格納されたユーザ定義メソッドのコードを記憶するコード記憶部と、コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法が提供される。

【００２３】本発明の他の１態様においては、ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューのエントリを有する部品プロパティテーブルと、ユーザ定義メソッド名とジェネリック・メソッド関数をプログラム開発支援システムから受けることにより、メソッドのコードをポイントするメソッドコールアダプタとを含む、実行時において、自ら管理するメソッドとプロパティを提供し、プログラム開発支援のためにプログラム開発支援システムにロードされるプログラム部品を格納する記憶媒体が提供される。

【００２４】本発明の他の１態様においては、ユーザによるプログラム部品の特定を検出することをデータ処理システムに指示するプログラムコード手段と、特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリとメソッド名を入力するためのメソッド情報エントリを表示することをデータ処理システムに指示するプログラムコード手段と、プロパティ情報エントリに入力されたユーザ定義プロパティ名をプロパティ・テーブルに登録することをデータ処理システムに指示するプログラムコード手段と、メソッド情報エントリに入力されたユーザ定義メソッド名をメソッド・テーブルに登録することをデータ処理システムに指示するプログラムコード手段とを含むプログラム部品の機能を拡張するためのプログラムを格納する記憶媒体が提供される。

【００２５】本発明の他の１態様においては、プログラム部品名とプログラム部品のプロパティ名とプログラム部品のメソッド名を指定する記述を含むプログラムコードを受領することをデータ処理システムに指示するプログラムコード手段と、プログラムコードに含まれる部品名とプロパティ名から、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定することをデータ処理システムに指示するプログラムコード手段と、プログラム部品名と、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいてプログラム部品のプロパティ・バリューを特定することをデータ処理システムに指示するプログラムコード手段と、プログラムコードに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定することをデータ処理システムに指示するプログラムコード手段と、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行することをデータ処理システムに指示するプログラムコード手段と、プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領することをデータ処理システムに指示するプログラムコード手段と、ジェネリック・メソッド・コード・アクセス情報を基づいて、コード記憶部に格納されたユーザ定義メソッドのコードにアクセスすることをデータ処理システムに指示するプログラムコード手段と、ユーザ定義メソッドのコードを実行することを前記データ処理システムに指示するプログラムコード手段とを含むプログラム・コードを実行するためのプログラムを格納する記憶媒体が提供される。

【００２６】

【実施例】

Ａ．ハードウエア構成

以下、図面を参照して本発明の実施例を説明する。図１を参照すると、本発明を実施するためのハードウェア構成の概観図が示されている。システム１００は、中央処理装置（ＣＰＵ）１とメモリ４とを含んでいる。ＣＰＵ１とメモリ４は、バス２を介して、補助記憶装置として

のハードディスク装置１３とを接続してある。フロッピーディスク装置（またはＣＤ－ＲＯＭ、ＭＯ、ＭＤ、ＺＩＰ、ＤＶＤ等の各種記憶媒体の駆動装置）２０はフロッピーディスクコントローラ等の各種コントローラ１９を介してバス２へ接続されている。

【００２７】フロッピーディスク装置（またはＣＤ－ＲＯＭ等の各種媒体の駆動装置）２０には、フロッピーディスク（またはＣＤ－ＲＯＭ等の各種記憶媒体）が挿入され、このフロッピーディスク等やハードディスク装置１３、ＲＯＭ１４には、オペレーティングシステムと協働してＣＰＵ等に命令を与え、本発明を実施するためのコンピュータ・プログラムのコードを記録することができ、このコードは、メモリ４にロードされることによって実行される。このコンピュータ・プログラムのコードは圧縮し、または、複数に分割して、複数の媒体に記録することもできる。

【００２８】システム１００は更に、ユーザ・インターフェース・ハードウェアを備えたシステムとすることができ、ユーザ・インターフェース・ハードウェアとしては、例えば、入力をするためのポインティング・デバイス（マウス、ジョイスティック等）７またはキーボード６や、視覚データをユーザに提示するためのディスプレイ１２がある。また、パラレルポート１６を介してプリンタを接続することや、シリアルポート１５を介してモデムを接続することが可能であり、シリアルポート１５およびモデムまたは通信アダプタ１８を介して他のコンピュータと通信を行うことが可能である。

【００２９】従って、本発明は、通常のパーソナルコンピュータ（ＰＣ）、やワークステーションやこれらの組合せによって実施可能であることを容易に理解できるであろう。ただし、これらの構成要素は例示であり、その全ての構成成要素が本発明の必須の構成要素となるわけではない。例えば、図１は、スタンド・アロン環境のシステムを示しているが、クライアント／サーバ・システムとして本発明を実現し、サーバ・マシンに本発明のシステム１００を配置し、クライアント・マシンは、サーバ・マシンに対して、イーサネット、トークン・リングなどでＬＡＮ接続し、クライアント・マシン側には、各種入力のための入力装置と、プログラムの実行結果を見るための表示装置を配置してもよい。また、後述するプログラム部品ライブラリ１３０または、その一部のプログラム部品をサーバ側に配置し、その他をクライアント側に配置したり、ビルダ１００のみをクライアントに配置し、その他をサーバ側に配置する等、設計段階で自由に変更できる。

【００３０】オペレーティング・システムとしては、Ｗｉｎｄｏｗｓ（マイクロソフトの商標）、ＯＳ／２（ＩＢＭの商標）、ＡＩＸ（ＩＢＭの商標）上のＸ－ＷＩＮＤＯＷシステム（ＭＩＴの商標）などの、標準でＧＵＩマルチウインドウ環境をサポートするものが望ましいが、本発明は、後述する非グラフィカルオブジェクトのみをプログラム部品とすることも可能であるため、ＰＣ－ＤＯＳ（ＩＢＭの商標）、ＭＳ－ＤＯＳ（マイクロソフトの登録商標）などのキャラクタ・ベース環境でも実現可能であり、特定のオペレーティング・システム環境に限定されるものではない。

【００３１】Ｂ．システム構成

次に、図２のブロック図を参照して、本発明のシステム構成について説明する。プログラム部品１３１，１３３は、部品ライブラリ１３０に格納されている。このプログラム部品は、その部品のプロパティ及びメソッドに関する情報を管理している。この管理している情報に関しては、後に詳述する。本発明のプログラム開発支援装置（以下"ビルダ"と呼ぶ）はプログラム部品に関する情報を記憶する部品情記憶部１１０を含み、その情報は、部品情報編集部１０１により、プログラマが追加及び修正可能になっている。ビルダ１００は、さらに、プロパティ編集部１０３を有しており、このプロパティ編集部１０３を使用することによって、プログラム部品１３１，１３３のプロパティ・バリューを変更することができる。オブジェクト視覚表現部１０５は、グラフィカルなプログラム部品を表示する。

【００３２】また、プログラマは、コード編集部１０７を使用してプログラムのコーディングを行うことができ、コーディングされたプログラム・コードは、コード記憶部１０９に格納される。そして、インタプリタ１２０は、このコード記憶部１０９に格納されたプログラム・コードと、プログラム部品の情報に基づいてプログラムを実行する。本発明の好適な実施例においては、コード実行部１２０は、インタプリタにより実施されている。しかし、コード実行部１２０はインタプリタには限定される概念ではなく、コンパイラ等によっても実施可能である。なお、このブロック図は、発明の理解の容易化を図るため一定の機能を有する構成を１つのまとまりとして表現したものであり、例えば、部品情報編集部１０１とプロパティ編集部１０３を統合した部品情報編集部として実施したり、また、部品情報編集部１０１を、部品プロパティ情報編集部と部品メソッド情報編集部に分割して実施することもできる。

【００３３】本発明と従来技術との差異を明確化するために、図３において、本発明のシステム構成の一実施例における従来技術との差異部分（斜線部分）を示す。この図において、プログラム部品１３１は、本発明のプログラム部品であり、プログラム部品１３３は、従来のプログラム部品である。本発明のビルダは、従来のプログラム部品と本発明のプログラム部品を区別することなく取り扱うことができることを示している。また、部品情報編集部１０１が新たに追加され、部品情報記憶部１１０、インタプリタ１２０に変更が加えられていることが示されている。

【００３４】Ｂ１．部品情報記憶部プロパティ・テーブル
図４は、本発明の好適な実施例における、部品情報記憶部のプロパティ・テーブルを示す図である。この部品情報記憶部のプロパティ・テーブル（以下単に「プロパティ・テーブル」と呼び、後述する部品プロパティ・テーブルと区別する）は、プログラム部品の部品名に対応して存在する。プロパティ・テーブル１５０には図に示すように、プロパティ名１５５、プロパティタイプ１５６、Ｇｅｔ関数１５７、Ｐｕｔ関数１５８及びユーザ定義フラグ１５９を管理している。

【００３５】このＧｅｔ関数１５７、Ｐｕｔ関数１５８は、プログラム部品１３１，１３３にアクセスし、プログラム部品１３１，１３３からそのプロパティの値をＧｅｔし、または、プログラム部品１３１，１３３へ、プロパティの値をＰｕｔするための関数である。このＧｅｔ関数１５７、Ｐｕｔ関数１５８は、プログラム部品１３１，１３３の性質によって、Ｇｅｔ関数１５７のみ必要であり、Ｐｕｔ関数１５８は不要である場合（そのプログラム部品からデータをもらうだけ）や、Ｐｕｔ関数１５８のみ必要であり、Ｇｅｔ関数１５７は不要である場合（そのプログラム部品へデータを書き出すだけ）も存在する。本願明細書において、このＧｅｔ関数１５７、Ｐｕｔ関数１５８の総称として「プロパティ・アクセス関数」を用いる。

【００３６】プロパティ・テーブル１５０は、既存のプロパティ情報部分１５１と、ユーザ定義のプロパティ情報部分１５３に分かれている。既存のプロパティ情報部分１５１は、プログラム部品１３３がシステムにインストールされると、そのプログラム部品１３３の有しているプロパティ名、プロパティ・タイプ、Ｇｅｔ関数、Ｐｕｔ関数の情報がこのプロパティ・テーブル１５０にコピーされ登録される。

【００３７】この一方、ユーザ定義のプロパティ情報部分１５３は、部品情報編集部１０１でプログラマがプログラム部品を指定して、追加の登録を行うと、登録されたプロパティ名、プロパティ・タイプがセットされ、プロパティ・アクセス関数（Ｇｅｔ関数、Ｐｕｔ関数）が自動生成される。本明細書において、ユーザがプロパティ名をまたはメソッド名を指定して、定義したプロパティまたはメソッドを「ユーザ定義プロパティ」または「ユーザ定義メソッド」と呼び、その他従来のプロパティやメソッドを「既存のプロパティ」、「既存のメソッド」と呼ぶこととする。

【００３８】このプロパティ・アクセス関数は、本願発明の好適な実施例においては、"ＧｅｔＰｒｏｐｅｒｔｙ／ＰｕｔＰｒｏｐｅｒｔｙ"という、ユーザ定義プロパティ共通のプロパティ・アクセス関数がセットされる（以下、この複数のユーザ定義のプロパティ情報に共通のプロパティ・アクセス関数を「ジェネリック・プロパティ・アクセス関数」と呼ぶ）。

【００３９】図に示すように、既存のプロパティ情報部分１５１と、ユーザ定義のプロパティ情報部分１５３が共通の形式で管理されているため、プロパティ編集部１０３、プログラムビルダ１００、及びインタープリタ１２０は、既存のプロパティ情報部分１５１と同様にユーザ定義のプロパティ情報部分１５３を扱うことができる。但し、既存のプロパティの情報と、ユーザ定義のプロパティの情報を共通の形式で管理する必要は必ずしも必須ではなく、部品情報記憶部１１０のプロパティ・テーブル１５０にアクセスするプロパティ編集部１０３等に変更を加え、２種類のテーブルにアクセス可能にすることにより、本発明を実施することもできる。さらに、既存のプロパティの情報と、ユーザ定義のプロパティの情報をテーブルで管理する必要は必ずしもなく、プロパティ名をキーに、それに対応した情報にアクセス可能な形式で、情報が格納されていれば、本発明を実施することができる。

【００４０】本発明の好適な実施例においては、このユーザ登録により作成されたプロパティには、ユーザが定義したプロパティであることを示すフラグ１５９がセットされる。但し、かかるフラグは本発明の必須の構成要素ではなくシステムが既存のプロパティ情報であるか、ユーザ定義のプロパティ情報であるかを判別できればよい。かかる判別の手法としては、既存のプロパティ情報と、ユーザ定義のプロパティ情報の格納する位置を変え、その位置情報で区別する方法や、ユーザ定義のプロパティ情報のプロパティ名を既存のプロパティ情報のプロパティ名と区別できる名前（例えばプロパティ名の先頭または最後を「ＵＳＲ」にする）によって判別することも可能である。

【００４１】Ｂ２．部品情報記憶部メソッド・テーブル
図５は、本発明の好適な実施例における、部品情報記憶部のメソッド・テーブルを示す図である。この部品情報記憶部のメソッド・テーブル（以下単に「メソッド・テーブル」と呼ぶ）は、プロパティ・テーブル１５０と同様にプログラム部品の部品名に対応して存在し、既存のメソッド情報１６１と、ユーザ登録のメソッド情報１６３によって構成されている。メソッド・テーブル１６０の既存のメソッド情報１６１には図に示すように、メソッド名１６５、実装関数アドレス情報（本発明の好適な実施例においてはＣ言語が使用されている）１６７、メソッド・タイプ１６８、及びユーザ定義フラグ１６９を管理している。

【００４２】この実装関数アドレス情報１６７は、このメソッド・テーブルが管理しているプログラム部品において、そのメソッドが実行された場合に起動されるモジュールをポイントするためのアドレス情報を管理している。例えば、そのプログラム部品が、ボタン・アイコンの部品であり、メソッドが"Ｍｏｖｅ"であった場合、そ

のボタン・アイコンを移動するために実行されるプログラムモジュールをポイントするためのアドレス情報を管理している。この既存のメソッド情報の実装関数は、プログラム部品が管理しているため、実装関数アドレス情報は、プログラム部品１３１，１３３の実装関数をポイントとするためのアドレスである。メソッド・タイプ１６８は、そのメソッドの実行に必要なプロパティを定義している。具体的には、「引き数１、引き数２、．．．引き数ｎ／戻り値１、戻り値２、．．．戻り値ｍ」の形式で定義されている。メソッドは、その種類によって、引き数のみを有するもの、戻り値のみを有するもの、双方を有するもの、双方とも有していないものが存在する。

【００４３】ユーザ定義のメソッド情報１６３も同様に、メソッド名１６５、実装関数アドレス情報１６７、メソッド・タイプ１６８、及びユーザ定義フラグ１６９を管理している。このように、既存のメソッド情報部分１６１と、ユーザ定義のメソッド情報部分１６３が共通の形式で管理されているため、プロパティ編集部１０３、プログラムビルダ１００、及びインタープリタ１２０は、既存のメソッド情報部分１６１と同様にユーザ定義のメソッド情報部分１６３を扱うことができる。ユーザ定義のメソッド情報１６３において、この実装関数アドレス情報１６７は、１つのプログラム部品に対して定義された、複数のユーザ定義のメソッド情報に共通のものであり、後述するプログラム部品１３１のメソッド・コール・アダプタ１３７をポイントするためのアドレス情報を管理している。

【００４４】プログラム部品におけるプロパティ情報と同様に、既存のメソッド情報部分１６１は、プログラム部品１３３がシステムにインストールされると、そのプログラム部品１３３の有しているメソッド名、実装関数、メソッド・タイプの情報がこのメソッド・テーブル１６０にコピーされ登録される。この一方、ユーザ定義のメソッド情報部分１６３は、部品情報編集部１０１でプログラマがプログラム部品を指定して、追加の登録を行うと、登録されたメソッド名、メソッド・タイプがセットされ、実装関数が自動生成される。

【００４５】この実装関数は、本願発明の好適な実施例においては、"ＣａｌｌＭｅｔｈｏｄ"という、ユーザ定義メソッド共通のプログラム部品に存在する見せかけの実装関数をポイントするための情報がセットされる（以下、この複数のユーザ定義のメソッド情報に共通の実装関数アドレス情報を「ジェネリック・メソッド関数」と呼ぶ）。また、本発明の好適な実施例においては、このユーザ登録により作成されたメソッドには、ユーザが定義したプロパティと同様に、ユーザが定義したメソッドであることを示すフラグ１６９がセットされる。但し、かかるフラグも、プロパティ情報におけるフラグ１５９と同様に、本発明の必須の構成要素ではなくシステムが既存のメソッド情報であるか、ユーザ定義のメソッド情報であるかを判別できればよい。また、プロパティ情報において述べたように、既存のメソッド情報とユーザ定義のメソッド情報は、共通の形式で管理することは必須ではない。また、テーブルの形式で管理することも必須ではない。

【００４６】Ｂ３．プログラム部品プロパティ情報

次に、プログラム部品１３１で管理されている情報について説明する。プログラム部品１３１においても、部品情報記憶部１１０で管理されている情報と同様に、プログラム部品のプロパティに関する情報と、プログラム部品のメソッド情報に関する情報を管理している。ただし、部品情報管理部１１０において説明したように、プログラム部品は、その性質上、プロパティ情報のみ、またはメソッド情報のみを管理している部品も存在するため、この２種類の情報を同時に有していることは本願発明の必須の構成用件とはならない。

【００４７】図６は、本発明の好適な実施例における、プログラム部品のプロパティ情報（以下「部品プロパティ情報」という）およびプログラム部品のプロパティ・テーブル（以下部品情報記憶部１１のプロパティ・テーブルと区別するため、「部品プロパティ・テーブル」と呼ぶ）を示す図である。部品プロパティ情報１７１で管理されている情報は、部品情報記憶部１１０において説明した既存のプロパティ情報１５１に対応する情報であり、部品プロパティ・テーブル１７３で管理されている情報は、ユーザ定義のプロパティ情報１５３に対応する情報である。部品プロパティ情報１７１は、その名が示すように、本発明の好適な実施例においても、テーブルの形式で管理されていない。プログラム部品がシステム導入されるときに、プログラム部品は、この部品プロパティ情報１７１の内容をビルダ１００側に宣言するため、ビルダ１００は、その情報の内容や種類を認識することができる。

【００４８】部品プロパティ情報１７１は、部品情報管理部１１０の既存のプロパティにおけるプロパティ・テーブル１５１の情報とほぼ同様の情報を管理している。プロパティ・テーブル１５１と部品プロパティ情報１７１の管理する情報で異なる点は、部品プロパティ情報１７１においては、プロパティに対する実際のプロパティ・バリューの情報を管理していることである。このプロパティ・バリュー１７９は、プログラム部品１３１のインストール時にデフォルトでセットされている場合や、プロパティ編集部１０３によってその値が設定される場合とがある。

【００４９】この一方、部品プロパティ・テーブル１７３においては、プロパティ名１７５、タイプ１７７、およびプロパティ・バリュー１７９の情報が管理されている。このプロパティ・テーブル１７３の情報は、部品情報編集部１０１におけるプログラマの登録によってセッ

トされる。なお、プロパティ・バリュー１７９の情報は、プロパティ編集部１０３から入力することもできる。

【００５０】Ｂ４．プログラム部品メソッド情報

次に、メソッドに関する情報がプログラム部品１３１においてどのように管理されているかを説明する。図７は、本発明の好適な実施例における、プログラム部品のメソッド情報（以下単に「部品メソッド情報」と呼ぶ）を示す図である。部品メソッド情報１８０は、部品情報記憶部１１０におけるメソッド・テーブル１６０と同様に既存のメソッド情報のメソッド名１８５、メソッド・タイプ１８８を管理している。実装関数１８７は、部品情報記憶部１１０におけるメソッド・テーブル１６０と異なり、そのメソッドが実行された場合に起動されるモジュールを管理している。

【００５１】ユーザ定義のメソッド情報に関しては、プログラム部品１３１には、既存のメソッド情報のようにプログラム部品側で管理されておらず、この代替として、メソッド・コール・アダプタ１３７が存在する。メソッド・コール・アダプタの具体的な内容は後述するが、ユーザ定義のメソッド情報のメソッドが実行された場合に起動されるモジュールをポイントするための関数が管理されている。

【００５２】Ｃ．システムの動作

本発明の好適な実施例においては、プログラム部品の機能を拡張するために、図３で示したように、部品情報記憶部１１０、プログラム部品１３１、インタプリタ１２０をそれぞれ拡張し、新たに部品情報編集部１０１を追加している。以下では、部品情報のユーザ定義、ビルダーによる部品情報の利用、実行時における部品情報の利用、という観点から各部分の拡張およびシステムの動作について説明する。

【００５３】Ｃ１．部品情報のユーザ定義

図８は、本発明の好適な実施例における、部品情報をプログラマ（ユーザ）が定義する際のデータの流れを示す図である。部品情報の更新、すなわち、プロパティの追加、変更、削除、及び、メソッドの追加、変更、削除は、部品情報編集部１０１が提供する機能によって行うことができる。

【００５４】（１）．プロパティの定義

プログラマは、部品情報編集部１０１において、プログラム部品の指定を行った後、プロパティ情報を入力するためのエントリを表示させることができる。本発明の好適な実施例においては、プログラム部品は、新規プログラム部品登録用アイコン、または既存のプログラム部品アイコンとして提供されているため、そのプログラム部品のアイコンをクリックすることにより選択することができる。

【００５５】プログラマが指定したプログラム部品がすでに存在している場合（本発明の好適な実施例においては、既に登録されているプログラム部品はプルダウンメニューにて選択可能になっている）は、そのプログラム部品のプロパティ情報、がユーザに修正及び削除可能に表示される。プログラマは、プロパティ・タイプ１５６を修正することができ、また、プロパティ名１５５を修正すると、修正後の新たなプロパティとして登録される。プログラマが選択したプログラム部品が新規なものである場合、プログラマは、プロパティ名１５５、プロパティ・タイプ１５６を入力する。本発明の好適な実施例においては、この更新（追加／修正／削除）に際し、シンタックスチェック等、内部的に矛盾の生ずる更新をチェックし、矛盾の生ずる更新を自動訂正またはリジェクトすることができる。

【００５６】このユーザによる登録操作により既存のプロパティ情報の更新の場合、内部的には部品情報記憶部１１０内のプロパティ・テーブル１５３のプロパティ情報が更新され（既存のプロパティ情報の更新の場合、プロパティ・タイプ１５６が更新され、既存のプロパティの削除の場合、そのプロパティのプロパティ名１５５、プロパティ・タイプ１５６、プロパティ・アクセス関数１５７，１５８、フラグ１５９が削除される）、さらに対応するプログラム部品１３１内の部品プロパティ情報１７１の情報もこれに対応して変更される。

【００５７】プログラマが指定したプログラム部品がシステムに存在していない新規なものである場合は、ユーザの指定したプログラム部品名が部品情報記憶部１１０と、対応するプログラム部品に登録され、登録されたプログラム部品のプロパティ・テーブル１５３にユーザが定義したプロパティ名１５５、プロパティ・タイプ１５６が登録され、プロパティ・アクセス関数１５７，１５８にジェネリック・プロパティ・アクセス関数が自動的にセットされ、フラグ１５９にユーザ定義プロパティであることを判別するための"１"が自動的にセットされる。そして対応するプログラム部品１３１内の部品プロパティ・テーブル１７３のプロパティ名１７５、プロパティ・タイプ１７７にユーザの定義したプロパティ名、プロパティ・タイプにセットされる。

【００５８】（２）．メソッドの定義

プロパティの定義と同様に、プログラマは、部品情報編集部１０１において、プログラム部品の指定を行うと、メソッド情報を入力するためのエントリが表示される。プログラマが指定したプログラム部品がすでに存在している場合は、そのプログラム部品のメソッド情報がユーザに修正及び削除可能に表示される。プログラマは、メソッド・タイプ１６８を修正することができ、また、メソッド名１６５を修正すると、修正後の新たなメソッドとして登録される。プログラマが選択したプログラム部品が新規なものである場合、プログラマは、メソッド名１６５、メソッド・タイプ１６８を入力する。本発明の好適な実施例においては、この更新（追加／修正／削

除）に際し、シンタックスチェック等、内部的に矛盾の生ずる更新をチェックし、矛盾の生ずる更新を自動訂正またはリジェクトすることができる。

【００５９】このユーザによる登録操作により、既存のメソッド情報の更新の場合、内部的には部品情報記憶部１１０内のメソッド・テーブル１６３のメソッド情報が更新され（既存のメソッド情報の更新の場合、メソッド・タイプ１６８が更新され、既存のメソッドの削除の場合、そのメソッドのメソッド名１６５、実装関数アドレス情報１６７、メソッド・タイプ１６８フラグ１６９が削除される）、さらに対応するプログラム部品１３１内の部品メソッド情報１８０の情報もこれに対応して変更される。

【００６０】プログラマが指定したプログラム部品がシステムに存在していない新規なものである場合は、ユーザの指定したプログラム部品名が部品情報記憶部１１０と、対応するプログラム部品に登録され、登録されたプログラム部品のメソッド・テーブル１６３にユーザが定義したメソッド名１６５、メソッド・タイプ１６８が部品情報編集部１０１で定義された内容にセットされ、実装関数アドレス情報１６７にジェネリック・メソッド関数が自動的にセットされ、フラグ１５９にユーザ定義メソッドであることを判別するための"１"が自動的にセットされる。この点において、部品情報編集部１０１と、部品情報記憶部１１０の動きは、プロパティ情報の場合と同様であるが、メソッド情報の場合は、本発明の好適な実施例においては、部品情報記憶部１１０内のメソッド・テーブル１６３のメソッド情報のみが更新され、プログラム部品１３１に対しての更新はない。これは、後述するメソッド・コール・アダプタの働きによって、他の場所に格納されているメソッドのコードを利用できるからである。但し、プログラム部品において、十分な記憶域が確保できるのであれば、プロパティの場合と同様に、プログラム部品側でメソッドのコードを管理することも可能である。

【００６１】Ｃ２．ユーザ定義部品情報によるプログラム作成支援

次に、このようにユーザによって定義されたプログラム部品がプログラム作成においてどのように利用されるかを説明する。図９は、本発明の好適な実施例における、ビルダがプログラム部品情報を利用する際のデータの流れを示す図である。

【００６２】（１）．プロパティ・バリューの設定

プログラマは、プロパティ編集部１０３を用いて、指定したプログラム部品のプロパティ・バリューを設定することができる。本発明の好適な実施例においては、部品情報記憶部１１０のプロパティ・テーブル１５０は、既存のプロパティとユーザ定義のプロパティを共通のテーブルで管理しているため、プロパティ編集部１０３は、従来のプロパティ編集部の機能に変更を加えなくても、部品情報記憶部１１０のユーザ定義のプロパティ・テーブル１５３のプロパティ情報にアクセスすることができる。但し、部品情報記憶部１１０のプロパティ・テーブル１５０は、既存のプロパティ・テーブル１５１とユーザ定義のプロパティ・テーブル１５３を共通の形式で管理することは必須ではなく、プロパティ編集部１０３を、別々で異なる既存のプロパティ・テーブル１５１とユーザ定義のプロパティ・テーブル１５３の２つのテーブルにアクセスできるようにプロパティ編集部の機能を拡張する事によって実施することができる。

【００６３】プログラマが定義したプロパティ・バリューが既存のプロパティ情報であった場合、ビルダ１００は、プロパティ・テーブル１５１のプロパティ・アクセス関数１５７，１５８を使用し、従来技術による手法を用いて、プログラム部品１３１の部品プロパティ情報１７１のプロパティ・バリュー１７９にアクセスし、プロパティ編集部１０３を用いて、指定したプログラム部品のプロパティ・バリューを設定することができる。

【００６４】プログラマが指定したプロパティ・バリューがユーザ定義プロパティのプロパティ・バリューであった場合、ビルダ１００は、部品情報記憶部１１０のプロパティ・テーブル１５３に格納されているプロパティ名１５５とジェネリック・プロパティ・アクセス関数１５７，１５８を用いて、プログラム部品１３１の部品プロパティ・テーブル１７３のプロパティ・バリュー１７９にアクセスし、プロパティ編集部１０３を用いて、プロパティ・バリューを設定することができる。

【００６５】具体的に説明すると、ビルダ１００は、まず、ユーザ定義のプロパティ・バリューをプロパティ編集部１０３で登録可能にするために、部品情報記憶部１１０のプロパティ・テーブル１５０に格納されているプロパティ名１５５を表示する。また、これに対応するプロパティ・バリューを表示するために、まず、そのプロパティが既存のプロパティであるか、ユーザ定義のプロパティであるかをプロパティ・テーブル１５０のフラグ１５９を検査することによって判断する。

【００６６】ユーザ定義のプロパティであると判断された場合は、ジェネリック・プロパティ・アクセス関数のうち、"ＧｅｔＰｒｏｐｅｒｔｙ"１５７を用いて、プログラム部品１３１の部品プロパティ・テーブル１７３にアクセスする。そして、部品プロパティ・テーブル１７３のプロパティ・バリュー１７９をプロパティ名１７５をキーに検索し、その値をプロパティ編集部１０３に渡すことによって、プロパティ・バリュー１７９をプロパティ編集部１０３に表示可能にする（登録された値がない場合は、空白が表示される）。

【００６７】そして、プログラマが、プロパティ編集部１０３のプロパティ・バリューを変更すると、ビルダ１００は、部品情報記憶部１１０のプロパティ・テーブル１５３に格納されているプロパティ名１５５とジェネリ

ック・プロパティ・アクセス関数のうち、"ＰｕｔＰｒｏｐｅｒｔｙ"１５８を用いて、プログラム部品１３１の部品プロパティ・テーブル１７３のプロパティ・バリュー１７９にアクセスし、プロパティ・バリューを更新する。

【００６８】（２）．メソッド情報の利用

プログラマはコード編集部１０７において、ユーザ定義プログラム部品のメソッドを特定する記載をすることができ、これにより、コード実行部１２０において実行することができる。詳しくは後に詳述する。

【００６９】Ｃ３．ユーザ定義部品情報を使用して作成されたプログラムの実行

次に、ユーザによって定義されたプログラム部品を使用して作成されたプログラムがどのように実行されるかを説明する。

【００７０】図１０は、本発明の好適な実施例における、プログラム実行時でのプログラム部品情報を利用する際のデータの流れを示す図である。まず、既存（ユーザ定義の情報でなく、従来のプログラム部品の情報）のプロパティ情報、メソッド情報の呼出については、インタープリタ１２０がコード記憶部１０９に記述されたコードを解釈し、ここで、プログラム部品名とプロパティ名、または、プログラム部品名とメソッド名が指定されていると、インタプリタ１２０は、プログラム部品の情報を利用することができる。

【００７１】（１）プロパティ情報の利用

図１１は、本発明の好適な実施例における、プログラム実行時でのユーザ定義プロパティ情報を利用する際のデータの流れを示す図である。インタプリタ１２０がプログラム部品のプロパティ情報を利用する場合、インタプリタは、ビルダ１００に対し、プログラム部品１３１のプロパティ・バリュー１７９にアクセスするための情報を求める。これに対し、ビルダ１００は、部品情報記憶部１１０のプロパティ・テーブル１５０をインタプリタ１２０から渡されたプロパティ名をキーに検索する。ビルダ１００は、インタプリタ１２０が要求するプロパティが、ユーザ定義のプロパティ情報か既存のプロパティ情報であるかを、部品情報記憶部１１０のプロパティ・テーブル１５０のフラグ１５９を検査することにより判断する。

【００７２】既存のプロパティ情報であった場合は、ビルダ１００は、インタプリタ１２０に、プロパティ・テーブル１５０のプロパティ・アクセス関数１５７，１５８を返す。このとき、インタプリタ１２０は、コードの内容から、プログラム部品のプロパティ・バリューへのアクセスがＧｅｔであるかＰｕｔであるかを判断することができるため、インタプリタ１２０は、ビルダ１００に対し、Ｇｅｔプロパティ・アクセス関数１５７、Ｐｕｔプロパティ・アクセス関数１５８のいずれかを指定して受領する方式と、Ｇｅｔプロパティ・アクセス関数１５７と、Ｐｕｔプロパティ・アクセス関数１５８の両方を受領し、そのいずれか一方を選択する方式とがある。

【００７３】インタプリタ１２０は、この情報によって、プログラム部品１３１，１３３の部品プロパティ情報にアクセスし、プロパティ・バリュー１７９にアクセスし、Ｇｅｔプロパティ・アクセス関数１５７の場合は、プロパティ・バリュー１７９の読み取りを行い、Ｐｕｔプロパティ・アクセス関数１５８の場合は、プロパティ・バリュー１７９への書き込みを行う。

【００７４】この一方、ユーザ定義のプロパティである場合は、ビルダ１００は、インタプリタ１２０に、プロパティ名を引き数としたジェネリックプロパティ・アクセス関数１５７，１５８を返す。このときも既存のプロパティ情報へのアクセスと同様に、インタプリタ１２０は、コードの内容から、プログラム部品のプロパティ・バリューへのアクセスがＧｅｔであるかＰｕｔであるかを判断することができるため、インタプリタ１２０は、ビルダ１００に対し、Ｇｅｔジェネリック・プロパティ・アクセス関数１５７、Ｐｕｔジェネリック・プロパティ・アクセス関数１５８のいずれかを指定して受領する方式と、Ｇｅｔジェネリック・プロパティ・アクセス関数１５７と、Ｐｕｔジェネリック・プロパティ・アクセス関数１５８の両方を受領し、そのいずれか一方を選択する方式とがある。

【００７５】また、インタプリタ１２０が、ビルダ１００に対し、プロパティ名をキーに、プロパティがユーザ定義プロパティであるか、既存のプロパティであるかの情報をもとめ、ビルダ１００は、プロパティ・テーブル１５０のフラグ１５９のみをインタプリタ１２０に返し、インタプリタ１２０側でジェネリック・プロパティ・アクセス関数を発行してもよい。さらに、プログラム部品１３１へのプロパティ情報登録時に、プログラム部品名、プロパティ名をインタプリタ１２０に渡すことにより、インタプリタ１２０は、ビルダ１００側にユーザ定義のプロパティか、既存のプロパティかの情報の問い合わせを行う必要はなくなる。

【００７６】インタプリタ１２０は、この情報によって、プログラム部品１３１の部品プロパティ・テーブルにアクセスし、また、プロパティ・バリュー１７９にアクセスし、Ｇｅｔジェネリック・プロパティ・アクセス関数１５７の場合は、プロパティ・バリュー１７９の読み取りを行い、Ｐｕｔジェネリック・プロパティ・アクセス関数１５８の場合は、プロパティ・バリュー１７９への書き込みを行う。

【００７７】（２）メソッド情報へのアクセス

図１２は、本発明の好適な実施例における、プログラム実行時でのユーザ定義メソッド情報を利用する際のデータの流れを示す図である。既存のメソッドコールは部品のメソッドに対応する関数を直接呼び出すことにより行われる。しかしながら、新たに追加されたメソッドに対

応する関数は、プログラム部品側に存在せず、インタプリタ１２０が管理している。そこで、ダミーの関数を用意し、インタープリタからこの関数を関数名とともに呼び出すことにする。インタプリタは、プロパティ情報とほぼ同様な方式に従って、メソッド名を引き数としたジェネリック・メソッド関数を発行する。

【００７８】プログラム部品１３１のメソッド・コール・アダプタ１３７は、インタプリタ１２０から受領したジェネリック・メソッド関数を、インタプリタ１２０で管理するメソッドのコード１２３を格納する位置をポイントする情報（以下「ジェネリック・メソッド・コード・アクセス情報」と呼ぶ）に変換する。本発明の好適な実施例においては、ジェネリック・メソッド・コード・アクセス情報は、メソッド名を引き数として、インタプリタ側に返される。本発明の好適な実施例においては、メソッドのコード１２３は、コード編集部１０７によって作成され、メソッド名をキーに、インタプリタ１２０が検索可能な状態でコード記憶部１０９に格納されており、インタプリタ１２０は、ジェネリック・メソッド・コード・アクセス情報を受領すると、メソッド名をキーにメソッドのコード１２３を検索し、メソッドのコード１２３を実行する。

【００７９】Ｄ．プログラマ作業手順

次に、本願発明の好適な実施例におけるプログラム部品の利用形態について、プログラマの作業手順に沿って説明する

【００８０】Ｄ１．部品情報定義手順

図１５乃至１９は、本発明の好適な実施例における、プログラム部品の設定手順を示す図であり、図１５は、プログラム部品の設定手順における初期状態を示す図である。図において、メインウインドウ３１０は、ビルダのメインウインドウであり、アプリケーション・ウインドウ３０２に対応する各種フォームや、コード記憶部１０９に格納された各種コードを管理するためのウインドウである。

【００８１】アプリケーション・ウインドウ３２０は、図２，３に示すオブジェクト視覚表現部１０５に対応するウインドウである。初期状態において、何も情報が入っておらず、プログラマはこのアプリケーション・ウインドウ３２０を用いてプログラム部品の指定等を行う。このアプリケーション・ウインドウによって、プログラマは直感的に理解しやすいプログラミングを行うことが可能になる。但し、後述する部品ライブラリの表示機能を拡張することによってもプログラム部品の指定を示すことが可能であるため、このアプリケーション・ウインドウは本発明の必須の構成要素ではない。

【００８２】図の左上に配置された複数のアイコンの集合体は部品ライブラリ３３０である。このアイコン１つ１つがプログラム部品に対応している。これらのアイコンのうち、ＵＤＯｂｊと記されたアイコン３３１はユーザ定義プログラム部品専用に提供されているアイコンである。プログラマが新たなプログラム部品を作成する場合、図１６に示すように、まず、アプリケーション・ウインドウ３２０に、ＵＤＯｂｊアイコン３３１をドロップする（アイコン３２１）。アプリケーション・ウインドウ３２０にアイコン３３１を位置づける手法は、ドラッグドロップの他、アイコン３３１をダブルクリックしたり、アイコン３３１を選択した状態において、予め設定されたキーをおす等、種々存在し、設計段階でさまざまに設定できる事項である。

【００８３】そして、図１７に示すように、プログラマがメイン・ウインドウ３１０のウインドウの項目をクリックし、プルダウンを表示させ、プロパティ・エディタの項目を選択すると、プロパティ・エディタ・ウインドウ３４０がオープンする。このアイコンに対応したウインドウをオープンさせる手法や手順は、例えば、アプリケーション・ウインドウ３２０に位置づけられたアイコン３２１をダブルクリックするとプロパティ・エディタ・ウインドウ３４０がオープンする等種々存在し、当業者は設計段階でそれらの手法や手順から所望のものを選択し実施化することができる。本発明の好適な実施例においては、新規登録用のアイコン３３１がアプリケーション・ウインドウ３２０にドロップされると、プロパティ・エディタ・ウインドウ３４０が自動的にオープンし、カーソルはプログラム部品名入力エントリ３４７に位置づけられ、強調表示によってプログラマにプログラム部品名の入力を促す。なお、このプロパティ・エディタ・ウインドウ３４０は図２，３のプロパティ編集部１０３に対応するウインドウである。

【００８４】プログラマは、まず、プログラム部品名の設定を行うことができる。図において、プログラマは、プロパティ・エディタ・ウインドウ３４０のプログラム部品名入力エントリ３４７にキーボード入力することにより、プログラム部品名を、"Ｏｒｄｅｒ"と設定している。そして、登録と記されたボタン・アイコン３４１をクリックすることによって、プログラマの入力の内容を登録する。この操作により、プログラム部品名"Ｏｒｄｅｒ"に対応して、部品情報記憶部１１０に空のプロパティ・テーブル１５０、メソッド・テーブル１６０が作成され、また、部品ライブラリ１３０に、プログラム部品名"Ｏｒｄｅｒ"の新たなプログラム部品が作成される。

【００８５】なお、本発明の好適な実施例においては、この初期登録状態において、部品情報記憶部１１０に登録されるプロパティ・テーブル１５０等は、空ではなく、どのアプリケーション・ウインドウに位置づけられたかを示す"Ｃｏｎｔａｉｎｅｒ"や、インデックス等のプロパティ名や、プロパティ・バリューが自動的にセットされ、部品情報記憶部１１０のプロパティ・テーブル１５０や、プログラム部品のプロパティ情報に登録され

る。このプログラム部品を部品ライブラリ３３０にアイコンとして表示し、選択可能な状態にすることも可能である。なおプロパティ・エディタ・ウインドウ３４０には、キャンセルのボタン・アイコン３４３が提供されており、このボタンがクリックされるとプロパティ・エディタ・ウインドウ３４０の各エントリに入力された未登録の文字列が無効にされる。

【００８６】プログラム部品名の設定が終わると、次にプログラマは、プロパティやメソッドの登録を行う。図１８において表示されているウインドウ３５０は、オブジェクト・ウインドウであり、図２，３における部品情報編集部に対応している。オブジェクト・ウインドウ３５０にはプログラム部品名３５５が表示されているため、プログラマは、プログラム部品"Ｏｒｄｅｒ"のプロパティまたはメソッドを定義していることを容易に知ることができる。図１８において、まず、プロパティの登録が行われている。プロパティの定義は、プロパティ入力エントリ３５７に、プロパティ名とプロパティ・タイプをキーボード入力することにより行うことができる。また、一覧と記されたボタン・アイコン３６１をクリックすると、定形的なプロパティ名とプロパティ・タイプがプルダウンメニューの形式で表示され、プログラマは、その一覧の中から１つを選択し、利用することができる。

【００８７】このプロパティ入力エントリ３５７に記載された文字列は、登録のボタン・アイコン３６３がクリックされることにより登録される。プロパティの登録がなされると、オブジェクト・ウインドウ３５０においては、登録されているプロパティのリスト３５９に登録された定義の内容が追加される。図においては、"ＰｒｏｄｕｃｔＩＤ　ａｓ　ｌｏｎｇ"というプロパティの定義文が登録されたことが示されている。

【００８８】このプロパティの登録がなされると、部品情報記憶部１１０のプロパティ・テーブル１５０のプロパティ名１５５，プロパティ・タイプが、ユーザの定義した内容に従って更新され、また、プロパティ・アクセス関数１５７，１５８のエントリにジェネリック・プロパティ・アクセス関数を示す情報がセットされる。また、フラグ１５９に、ユーザ定義のプロパティであることを示す「１」がセットされる。同様に、プログラム部品１３１においても、部品プロパティ・テーブル１７３のプロパティ名とプロパティ・タイプが登録される。なお、プロパティの登録のボタン・アイコンがクリックされたことを検出して、プロパティ・エディタ・ウインドウ３４０を登録後の内容で再表示させることも可能である。

【００８９】図１９において、プロパティの登録に次いでメソッドの登録が行われている。プロパティの登録とメソッドの登録は、どちらを先に行ってもよいが、本明細書においては、読者の負担軽減のため、プロパティを先に、メソッドを後に説明を行うこととする。メソッドの登録の、プロパティの登録と同様に、入力エントリ３６５に、メソッド名と、メソッド・タイプをキーボード入力することにより行うことができる。また、一覧と記されたボタン・アイコン３６９をクリックすると、定形的なメソッド名とメソッド・タイプがプルダウンメニューの形式で表示され、プログラマは、その一覧の中から１つを選択し、利用することができる。

【００９０】このメソッド入力エントリ３６５に記載された文字列は、登録のボタン・アイコン３６３がクリックされることにより登録される。メソッドの登録がなされると、オブジェクト・ウインドウ３５０においては、登録されているメソッドのリスト３６７に登録された定義の内容が追加される。図においては、"ＱｕｅｒｙＰｒｉｃｃ（ＩＤ　ａｓ　ｌｏｎｇ）　ａｓ　ｓｔｒｉｎｇ"というメソッドの定義文が登録されたことが示されている。

【００９１】このメソッドの登録がなされると、部品情報記憶部１１０のメソッド・テーブル１６０のメソッド名１６５，メソッド・タイプ１６８が、ユーザの定義した内容に従って更新され、また、実装関数アドレス情報１６７のエントリにジェネリック・メソッド関数を示す情報が自動的にセットされる。また、フラグ１６９に、ユーザ定義のメソッドであることを示す「１」がセットされる。

【００９２】Ｄ２．プログラム作成手順

次に、上述のユーザ定義のプログラム部品の情報が、プログラム開発時に、どのようにプログラマに用いられるかを説明する。図２０、２１は、本発明の好適な実施例における、プログラム部品の利用手順を示す図である。図２０において、プログラマは、ユーザ定義プロパティのプロパティ・バリューの設定を行っている。プログラマは、プロパティ・エディタ・ウインドウ３４０を再表示させると、ユーザ定義プロパティに対応したエントリが提供されているため、プロパティ・バリューを設定することができる。

【００９３】これは、プロパティ・エディタ・ウインドウ３４０の再表示が行われると、ビルダ１００は、部品情報記憶部１１０のプロパティ・テーブル１５０に格納されているプロパティ名１５５をすべて表示する。また、これに対応するプロパティ・バリューを表示するために、まず、そのプロパティが既存のプロパティであるか、ユーザ定義のプロパティであるかをプロパティ・テーブル１５０のフラグ１５９を検査する事によって判断する。ユーザ定義のプロパティである場合は、フラグ１５９の値が１であるため判断することができる。この場合、ジェネリック・プロパティ・アクセス関数のうち、"ＧｅｔＰｒｏｐｅｒｔｙ"１５７を用いて、プログラム部品１３１の部品プロパティ・テーブル１７３にアクセスする。そして、部品プロパティ・テーブル１７３

のプロパティ・バリュー１７９をプロパティ名１７５をキーに検索し、その値をプロパティ編集部１０３に渡すことによって、プロパティ・バリュー１７９をプロパティ編集部１０３に表示する。この段階において、ユーザ定義のプロパティのプロパティ・バリューは何も設定されていない状態なので、空白が表示されることとなる。

【００９４】そして、プログラマが、プロパティ名"Ｐｒｏｄｕｃｔ"に対し、プロパティ・バリュー"ＰＳ／５５"、（「ＰＳ／５５」はＩＢＭ社の商標）、プロパティ名"ＰｒｏｄｕｃｔＩＤ"に対し、プロパティ・バリュー"５４３３"を入力すると、ビルダ１００は、部品情報記憶部１１０のプロパティ・テーブル１５３に格納されているプロパティ名１５５とジェネリック・プロパティ・アクセス関数のうち、"ＰｕｔＰｒｏｐｅｒｔｙ"１５８を用いて、プログラム部品１３１の部品プロパティ・テーブル１７３のプロパティ・バリュー１７９にアクセスし、プロパティ・バリューを更新する。

【００９５】次に、コード編集部におけるプログラム開発時に、どのようにプログラマに用いられるかを説明する。図２１において、プログラマは、プログラムのコーディングをしている。プログラマは、このコード・エディタ・ウインドウ３８０において、ユーザ定義メソッドのコードの作成と、メインロジック部のコード作成を行うことができる。図において、ウインドウ３８０はコード・エディタ・ウインドであり、図２，３におけるコード編集部に対応している。この図において、プログラム部品名"Ｏｒｄｅｒ"，メソッド名"ＱｕｅｒｙＣｕｓｔｏｍｅｒ"のユーザ定義メソッドのコードを定義している。図におけるのコード記述エントリ３９３で記載されたコードの内容が、コード記憶部１０９に格納され、インタプリタ１２０に渡されることにより、ユーザ定義メソッドのコードとして実行されることとなる。ここに記載されているメソッドのコードの記述から、他のプログラム部品のプロパティやメソッドを使用するための記述と同様のコーディングで、メインロジックの記述を行うことができる。コード記述エントリ３９３には、図に示すように、他のプログラム部品の部品名３９５とプロパティ３９７を指定し、または、他のプログラム部品の部品名３９９とメソッド名４０１を引き数等を付加して指定することにより他のプログラム部品のプロパティやメソッドを利用することもできる。

【００９６】このコード・エディタのウインドウでは、モジュール・エントリ３８７、オブジェクト・エントリ３８９、メソッド・エントリ３９１の３種類のエントリが提供されている。プログラマはこれらのエントリに、直接キーボード入力することによって、コード記述エントリに入力したコードの部品名とメソッド名を指定する。各エントリ３８７，３８９，３９１には、プルダウンメニューの機能を有している。

【００９７】モジュール・エントリ３８７のプルダウン・メニューは、他のフォーム（コード・エディタで編集した他のコード）を呼び出すために設けられている。このプルダウン・メニューで選択されたフォームが、コード記述エントリ３９３に表示される。オブジェクト・エントリ３８９のプルダウン・メニューは、他のプログラム部品を検索し選択するためのプルダウン・メニューである。メソッド・エントリ３９１のプルダウン・メニューは、他のメソッドを検索し選択するためのプルダウン・メニューである。

【００９８】実行のためのボタン・アイコン３８１がクリックされると、このコードがインタプリタ１２０に渡され、前述の手順に従ってコードが実行される。なお、実行のためのボタン・アイコン３８１の他に中断、中止のためのボタン・アイコン３８３，３８５が提供されており、このアイコンをクリックすることにより、プログラムの実行を中断または中止することができる。

【００９９】

【発明の効果】以上説明したように、本発明によれば、部品情報記憶部において、プログラム部品が本来持っている機能に関する情報を蓄積するだけでなく、実際の部品にはない新たな機能も蓄積し、プログラム構築支援構築装置内の他の装置やインタプリタが利用できるようにする。

【０１００】また、本発明によれば、部品情報編集部において、プログラム部品に新たなプロパティならびにメソッドを追加することにより部品のみかけ上の機能を拡張することができる。

【０１０１】また、部品情報記憶部に保存された情報を基にして、プログラム部品が元々持っていたプロパティーと、新たに追加されたプロパティをプロパティ編集部で区別なく変更することができる。

【０１０２】そして、部品情報記憶部に保存された情報に基づいて、メソッドの骨格のコードを自動的に生成し、新たに追加されたメソッドをコード編集部で記述し、新たに追加されたメソッドやプロパティの利用に際して、部品が元々持っていたものと区別なく利用することができる。

【０１０３】さらに、プログラムの実行に関して、インタプリターからプログラム部品を呼び出す場合に、部品情報記憶部に保存された情報を用いて、新たに追加されたプロパティの参照・変更、ならびにメソッドを呼び出すことができる。

【０１０４】プログラム部品は、通常Ｃ言語等の言語によって記述されているため、従来は、あるプログラム部品に対して新たにプロパティやメソッドを追加することは、これと異なる言語であるベーシック言語等によっては行うことができず、Ｃ言語等によって行わなければならなかった。しかし、本発明のプログラム部品へのプロパティ情報やメソッド情報の追加方法を用いれば、ベーシック言語等のよりユーザにとって理解しやすい言語で

コーディングすることが可能となり、開発効率も向上することができる。

【０１０５】

【図面の簡単な説明】

【図１】　ハードウェア構成の一実施例を示すブロック図である。

【図２】　本発明のシステム構成の一実施例を示す機能ブロック図である。

【図３】　本発明のシステム構成の一実施例における従来技術との差異部分を示す機能ブロック図である。

【図４】　本発明の好適な実施例における、部品情報記憶部のプロパティ・テーブルを示す図である。

【図５】　本発明の好適な実施例における、部品情報記憶部のメソッド・テーブルを示す図である。

【図６】　本発明の好適な実施例における、プログラム部品のプロパティ・テーブルを示す図である。

【図７】　本発明の好適な実施例における、プログラム部品のメソッド・テーブルを示す図である。

【図８】　本発明の好適な実施例における、部品情報を更新する際のデータの流れを示す図である。

【図９】　本発明の好適な実施例における、ビルダが部品情報を利用する際のデータの流れを示す図である。

【図１０】　本発明の好適な実施例における、プログラム実行時での部品情報を利用する際のデータの流れを示す図である。

【図１１】　本発明の好適な実施例における、プログラム実行時での部品情報を利用する際のデータの流れを示す図である。

【図１２】　本発明の好適な実施例における、プログラム実行時での部品情報を利用する際のデータの流れを示す図である。

【図１３】　従来のプログラム部品の利用例を示す図である。

【図１４】　従来のプログラム作成支援システムを示す＊図である。

【図１５】　本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図１６】　本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図１７】　本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図１８】　本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図１９】　本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図２０】　本発明の好適な実施例における、プログラム部品の利用手順を示す図である。

【図２１】　本発明の好適な実施例における、プログラム部品の利用手順を示す図である。

【０１０６】

【符号の説明】

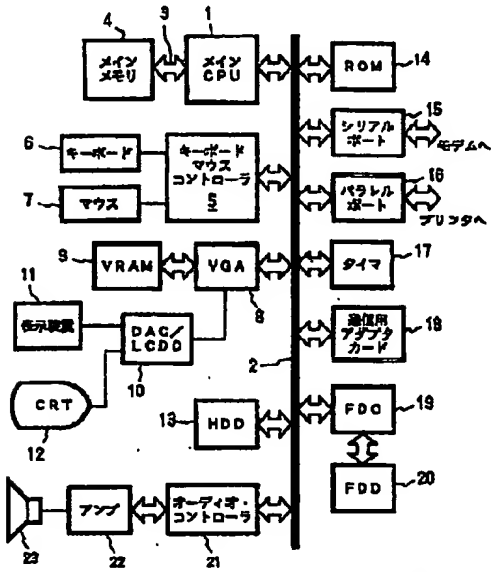
１００　ビルダー（プログラム開発支援装置）
１０１　部品情報編集部
１０３　プロパティ編集部
１０５　オブジェクト視覚表現部
１０７　コード編集部
１０９　コード記憶部
１１０　部品情報記憶部
１２０　インタプリタ
１３０　部品ライブラリ
１３１，１３３　プログラム部品
１３７　メソッドアダプタ
１５０　プロパティ・テーブル
１６０　メソッド・テーブル
１７１　部品プロパティ情報
１７３　部品プロパティ・テーブル
１８０　部品メソッド情報

【図４】

	名前 155	タイプ 156	Get関数 157	Put関数 158	フラグ 159	
151	Width	Integer	GetWidth	PutWidth	0	(通常のプロパティー)
	Label	String	GetLabel	PutLabel	0	
	⋮	⋮	⋮	⋮	⋮	
153	Aaa	Integer	GetPROPERTY	PutPROPERTY	1	(ユーザ定義のプロパティー)
	Bbb	String	GetPROPERTY	PutPROPERTY	1	
	⋮	⋮	⋮	⋮	⋮	

プロパティー・テーブル 150

【図１】



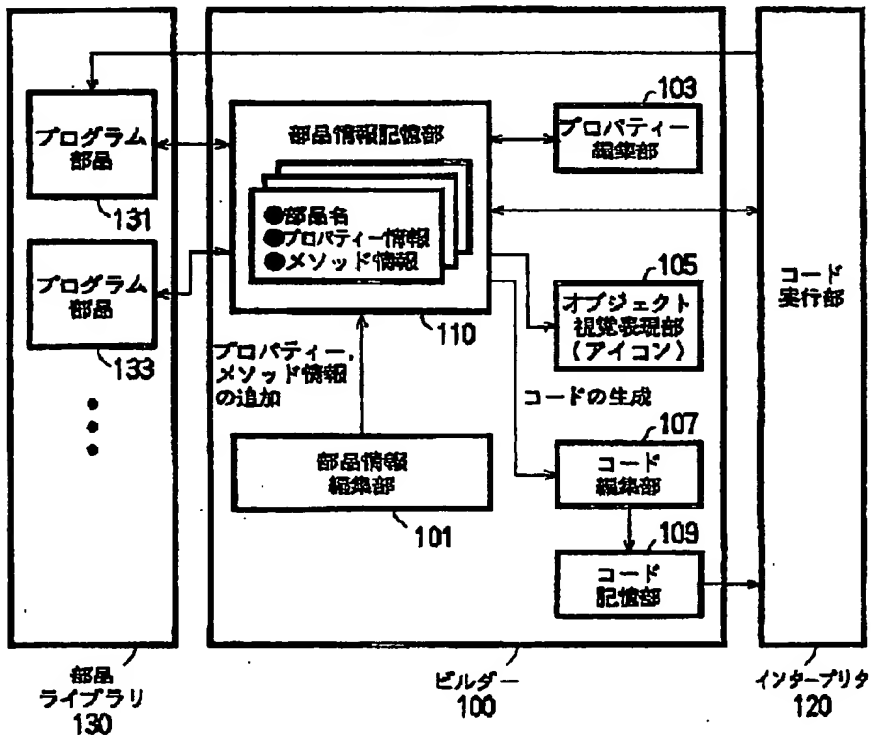
【図５】

名前 155	実装関数アドレス情報 157	タイプ 158	フラグ 159
Move Rename GetStatus	Moveアドレス情報 Renameアドレス情報 GetStatusアドレス情報	x as IFInteger, y as Integer/- a as String/- -/Return Code as Integer	0 0 0
Xxx Yyy	CallMethodアドレス情報 CallMethodアドレス情報	a as Integer a as String	1 1

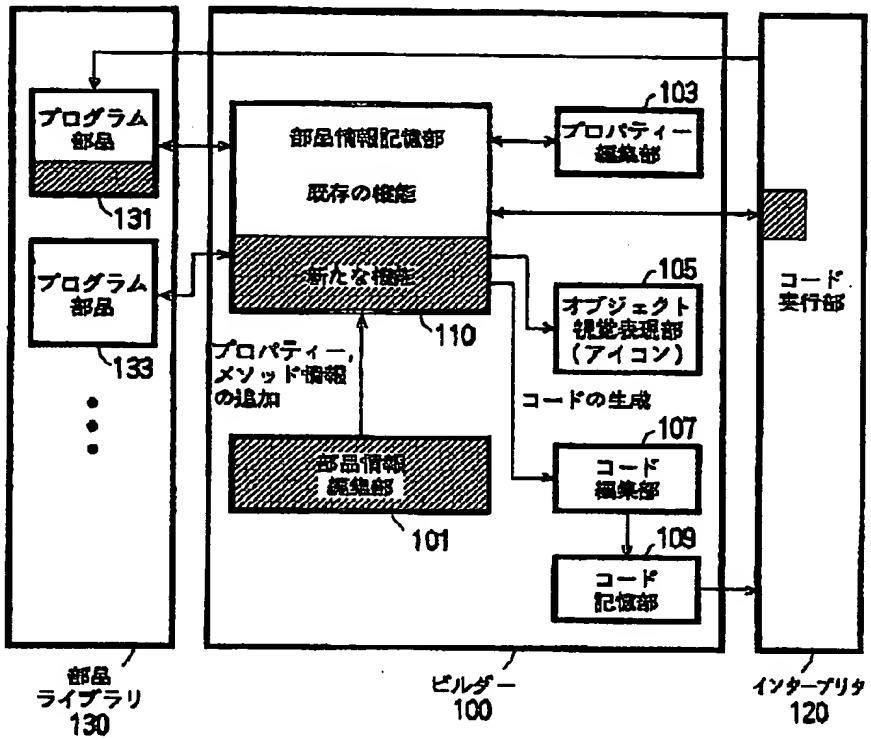
161　183

メソッド・テーブル 160

【図２】



【図３】



【図６】

名前 175	タイプ 176	GET関数 177	PUT関数 178	値 179
Width	Integer	GetWidth	PutWidth	値
Label	String	GetLabel	PutLabel	値
⋮	⋮	⋮	⋮	⋮

部品プロパティ情報 171

名前 175	タイプ 177	値 179
Aaa	Integer	値
Bbb	String	値
⋮	⋮	⋮

173

部品プロパティ・テーブル 173

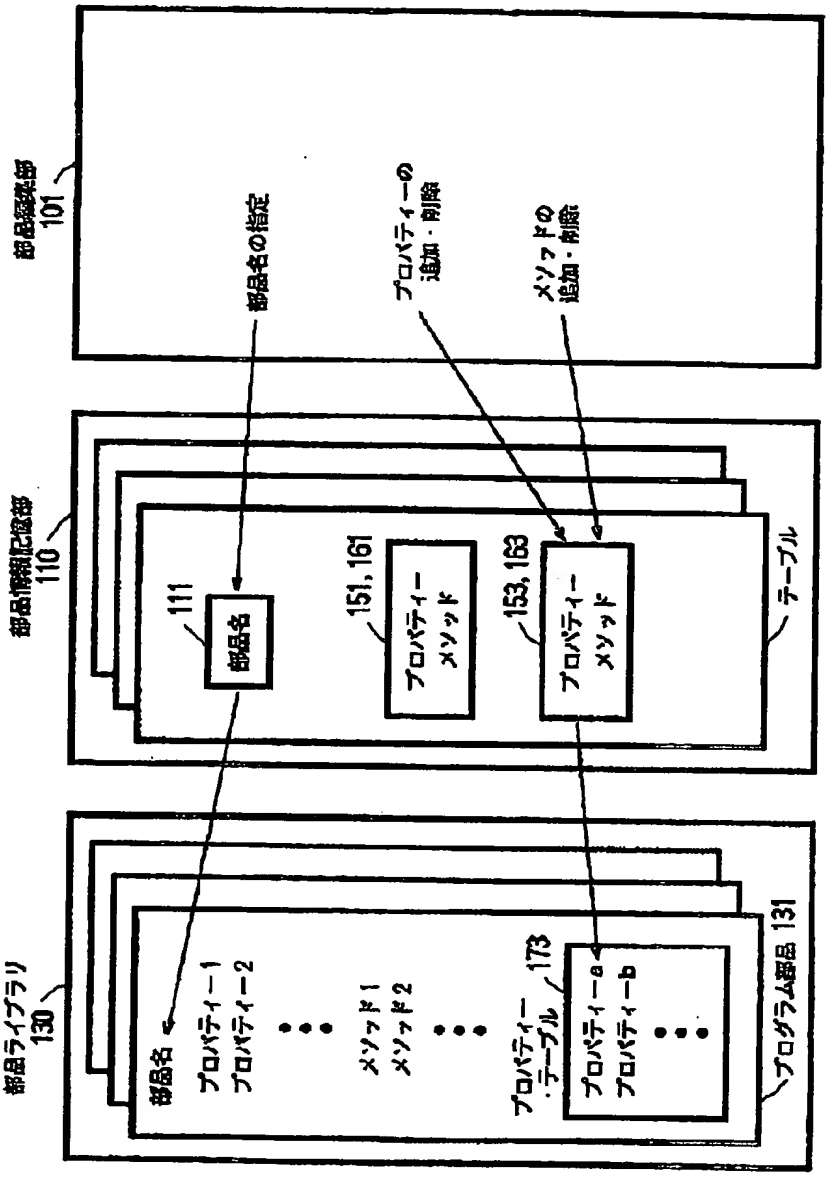
【図７】

名前 185	実装関数 187	タイプ 188
Move	Moveアドレス情報	x as Integer, y as Integer/-
Rename	Renameアドレス情報	a as String/-
GetStatus	GetStatusアドレス情報	-/ReturnCode as Integer
⋮	⋮	⋮

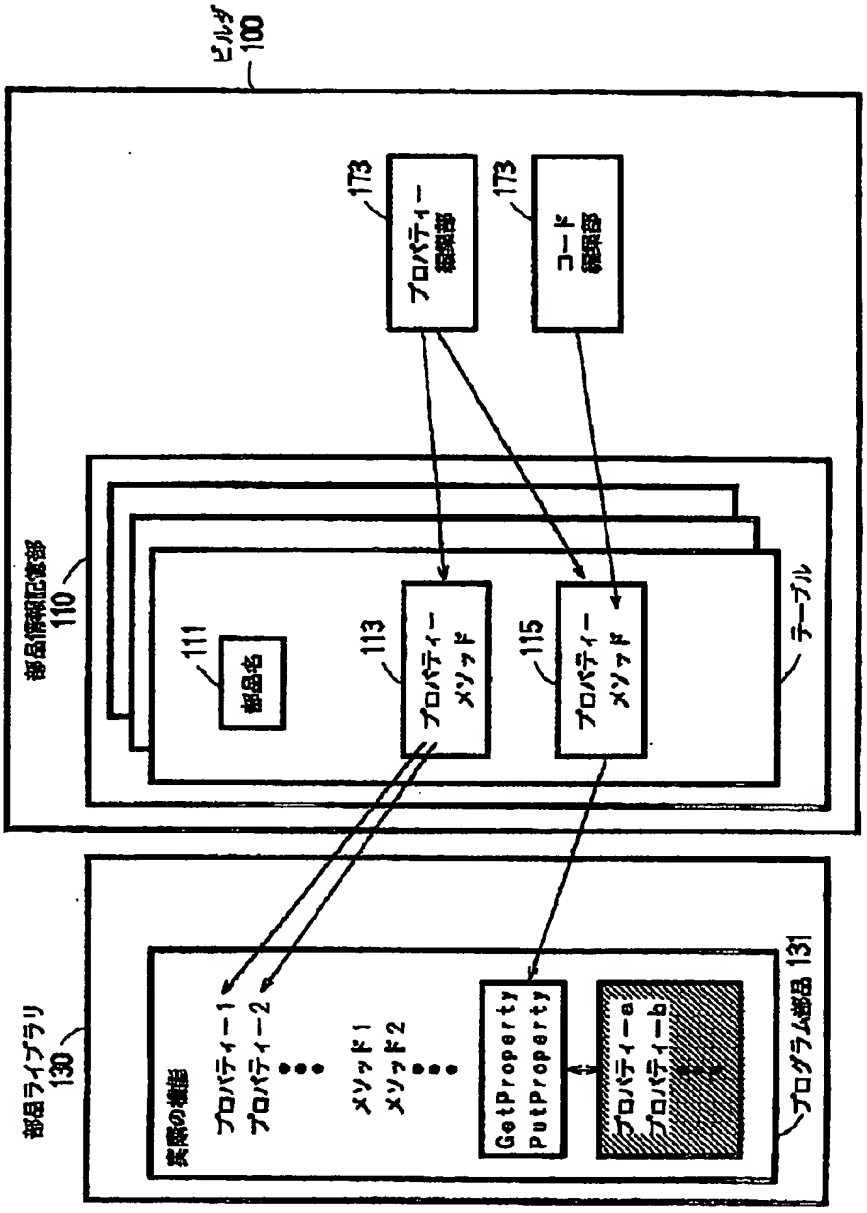
部品メソッド情報 180

メソッド・コール・アダプタ
187

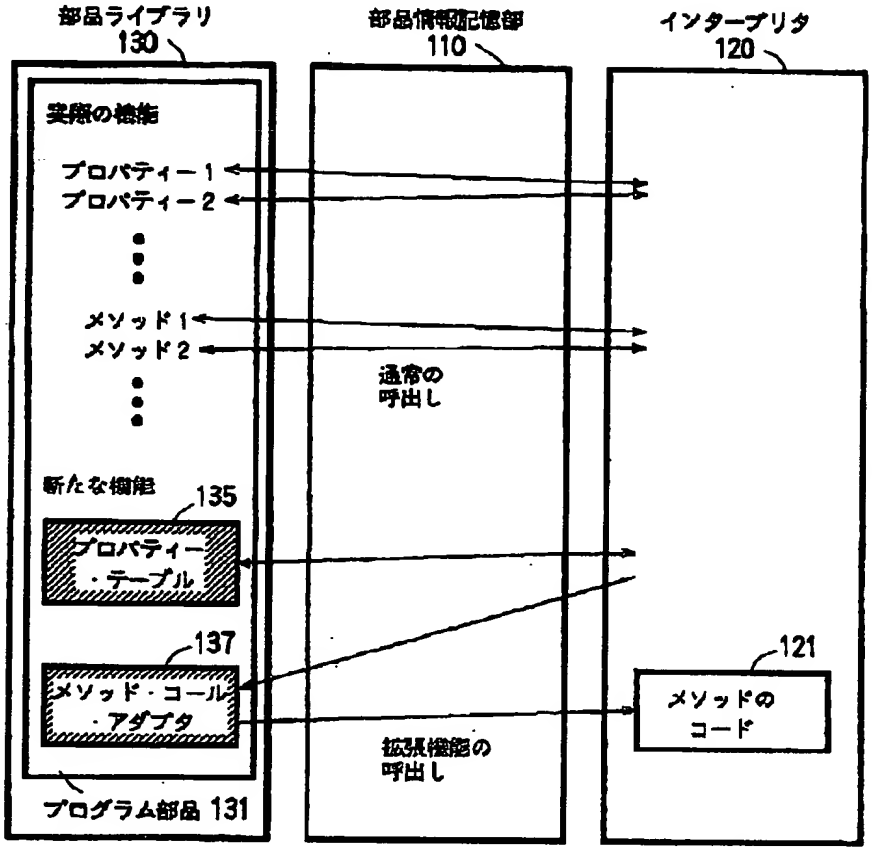
【図8】


部品編集部
101
部品名の指定
プロパティーの
追加・削除
メソッドの
追加・削除
部品情報記憶部
110
111
部品名
151, 161
プロパティー
メソッド
153, 163
プロパティー
メソッド
テーブル
部品ライブラリ
130
部品名
プロパティー1
プロパティー2
メソッド1
メソッド2
プロパティー
テーブル
173
プロパティーa
プロパティーb
プログラム部品 131

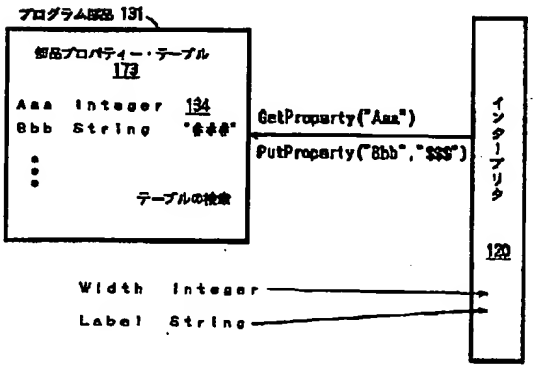
【図9】


ビルダ
100
173
プロパティー
173
コード
部品情報記憶部
110
111
部品名
113
プロパティー
メソッド
115
プロパティー
メソッド
テーブル
部品ライブラリ
130
実際の機能
プロパティー1
プロパティー2
メソッド1
メソッド2
GetProperty
PutProperty
プロパティーa
プロパティーb
プログラム部品 131

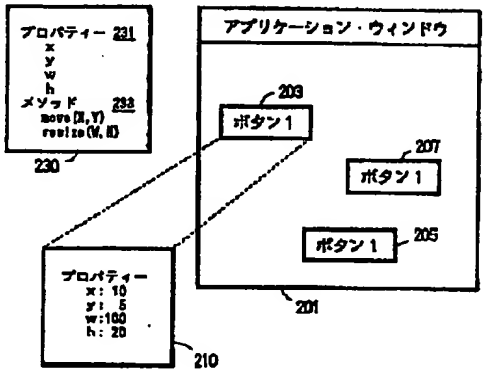
【図１０】


部品ライブラリ
130
部品情報記憶部
110
インタープリタ
120
実際の機能
プロパティー１
プロパティー２
メソッド１
メソッド２
通常の
呼出し
新たな機能
135
プロパティー
・テーブル
137
メソッド・コール
・アダプタ
121
メソッドの
コード
拡張機能の
呼出し
プログラム部品 131

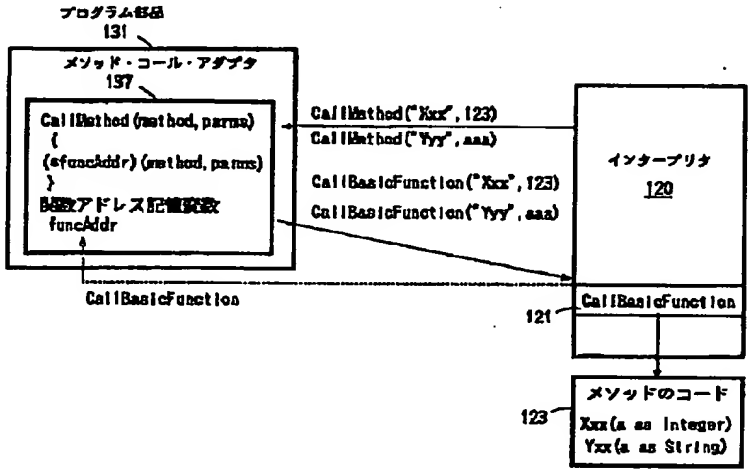
【図１１】


プログラム部品 131
部品プロパティー・テーブル
172
Aaa Integer 134
Bbb String "＊＊＊"
テーブルの検索
GetProperty("Aaa")
PutProperty("Bbb","$$$")
インタープリタ
120
Width Integer
Label String

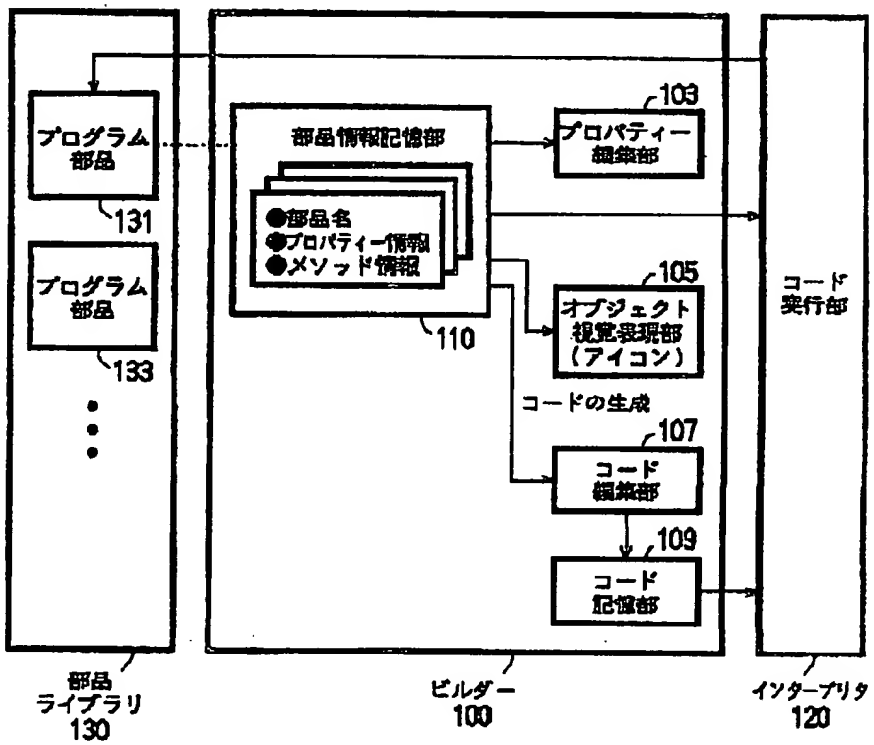
【図１３】


プロパティー 231
x
y
w
h
メソッド 232
move(x,y)
resize(w,h)
230
アプリケーション・ウィンドウ
208
ボタン１
207
ボタン１
205
ボタン１
プロパティー
x: 10
y: 5
w:100
h: 20
210
201

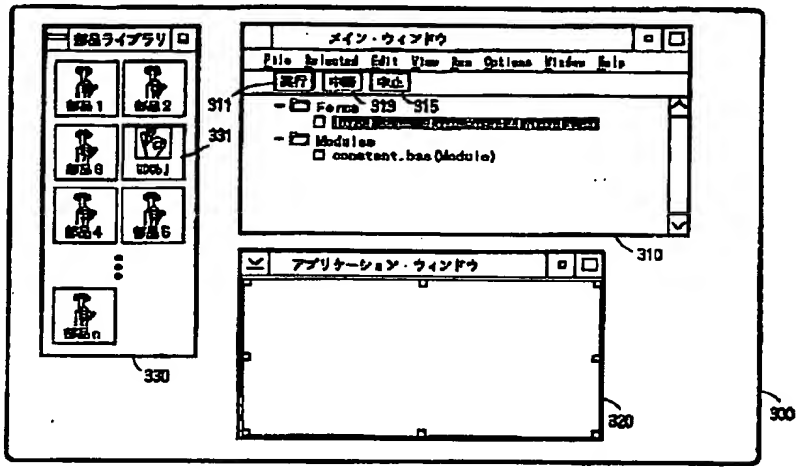
【図１２】


プログラム部品
131
メソッド・コール・アダプタ
137
CallMethod (method, parms)
{
(efuncAddr) (method, parms)
}
関数アドレス記憶変数
funcAddr
CallMethod("Xxx", 123)
CallMethod("Yyy", aaa)
CallBasicFunction("Xxx", 123)
CallBasicFunction("Yyy", aaa)
CallBasicFunction
インタープリタ
120
121
CallBasicFunction
メソッドのコード
123
Xxx(a as Integer)
Yxx(a as String)

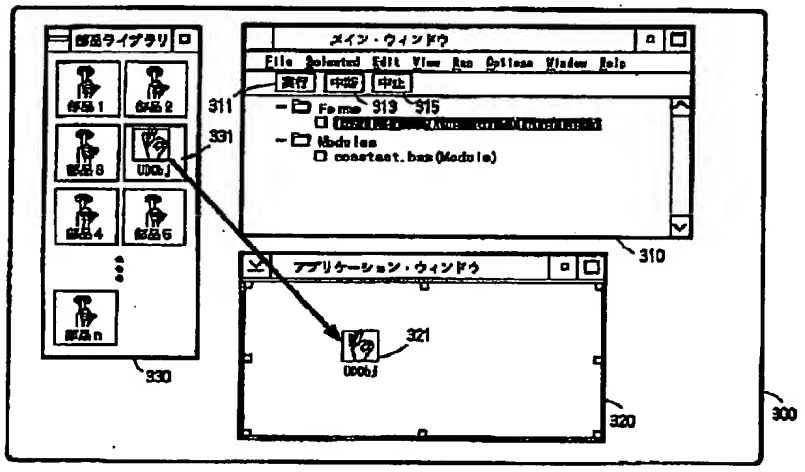
【図１４】


プログラム部品
131
プログラム部品
133
部品情報記憶部
●部品名
●プロパティー情報
●メソッド情報
110
103
プロパティー編集部
105
オブジェクト視覚表現部（アイコン）
コードの生成
107
コード編集部
109
コード記憶部
コード実行部
部品ライブラリ
130
ビルダー
100
インタープリタ
120

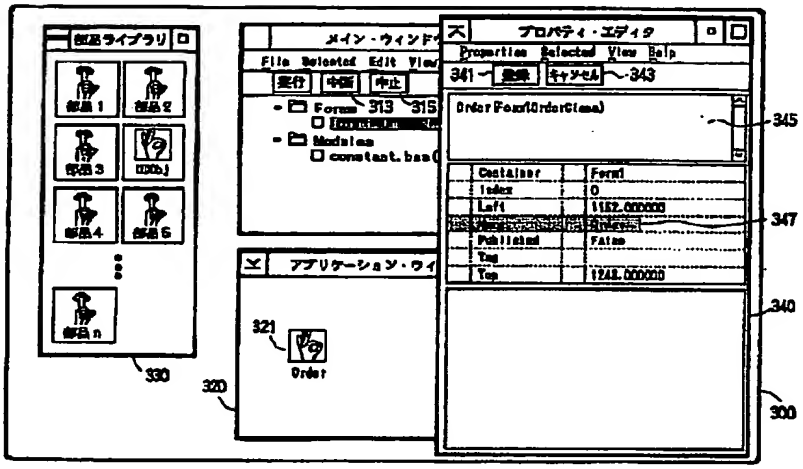
【図１５】



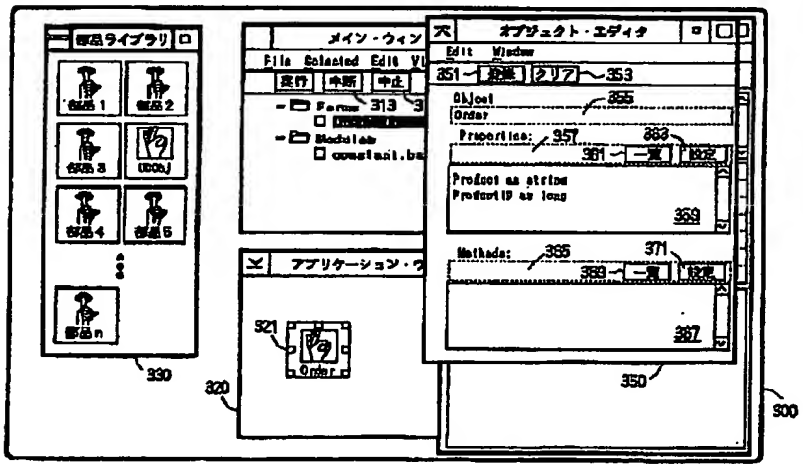
【図１６】



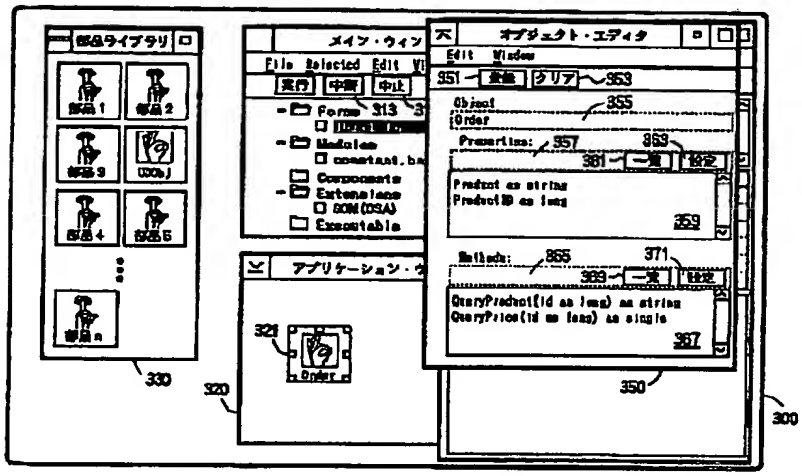
【図１７】


部品ライブラリ
部品1
部品2
部品3
(Obj)
部品4
部品5
部品n
330
メイン・ウィンドウ
File Selected Edit View
実行
中断
中止
313
315
Forms
Modules
constant.bas
アプリケーション・ウィ
321
Order
320
プロパティ・エディタ
Properties Selected View Help
341
登録
キャンセル
343
Order Form(OrderClass)
345
Container
Form
Index
0
Left
1182.000000
347
Published
False
Tag
Top
1748.000000
340
300

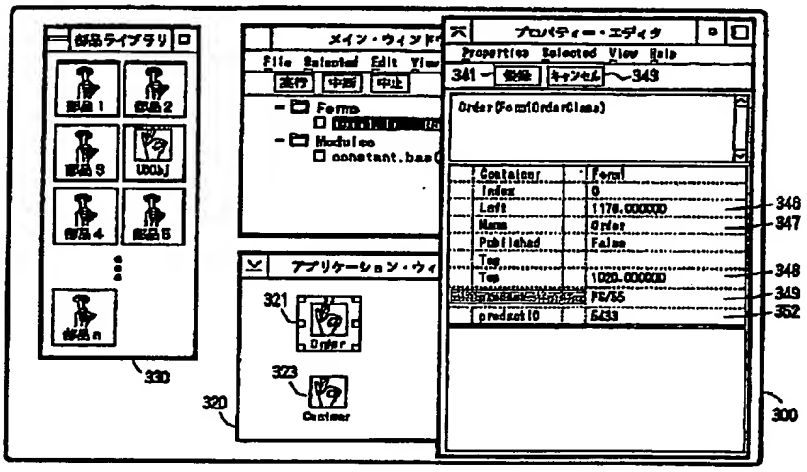
【図１８】


部品ライブラリ
部品1
部品2
部品3
(Obj)
部品4
部品5
部品n
330
メイン・ウィン
File Selected Edit
実行
中断
中止
313
Forms
Modules
アプリケーション・ウ
321
Order
320
オブジェクト・エディタ
Edit Window
351
登録
クリア
353
Object
355
Order
Properties:
357
363
361
一覧
設定
Product as string
ProductID as long
359
Methods:
365
371
369
一覧
設定
367
350
300

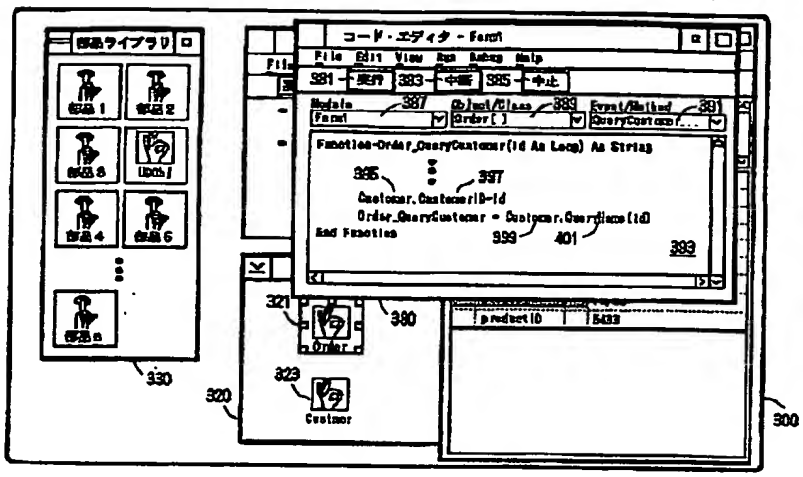
【図１９】


部品ライブラリ
メイン・ウィン
オブジェクト・エディタ
Edit Window
351
実行
クリア
353
Object
355
Order
Properties:
357
363
361
Product as string
Product as long
359
Methods:
365
371
369
QueryProduct(id as long) as string
QueryPrice(id as long) as single
367
アプリケーション・ウ
321
Order
330
320
350
300

【図２０】


部品ライブラリ
メイン・ウィンドウ
プロパティー・エディタ
Properties Selected View Help
341
実行
キャンセル
343
Order(Form!OrderClass)
Container
Form
Index
0
Left
1170.000000
Name
Order
Published
False
Top
1020.000000
predcat10
5433
348
347
348
349
352
アプリケーション・ウィ
321
Order
323
Customer
330
320
300

【図２１】



フロントページの続き

(72)発明者　中村　祐一
神奈川県大和市下鶴間1623番地11　日本アイ・ビー・エム株式会社　大和事業所内

(72)発明者　田胡　和哉
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社　大和事業所内